escola superior gallaecia

Mestrado Integrado em Arquitectura e Urbanismo

Agent-based Modelling and Swarming in Urban Architecture

Simulating Urban Environment

Tiago André Martins Gomes

Orientador: David Leite Viana

Vila Nova de Cerveira, Setembro de 2016

Contents

Agradecimentos

Esta dissertação é dedicada aqueles que incessantemente me apoiaram, motivaram e estiveram presentes nos bons e difíceis momentos deste percurso.

À minha famíla, ao meu pai e à minha mãe pela incansável força e motivação que me dedicaram nos momentos mais difíceis. À minha tia avó Etelvina Fernandes pela compreensão e apoio incondicional.

Ao meu orientador David Leite Viana pelo conhecimento partilhado e extrema capacidade motivacional que me proporcionou ao longo deste percurso.

Aos meus colegas Cátia Martins pelas longas conversas e partilha de conhecimento durante as viagens para Vila Nova de Cervira , Antón Tejada e Antón Iglésias Castro pelos imensos momentos de debate sobre os temas abordados. Ao Rui Costa pelas longas sessões de programação, exploração e troca de conhecimento que partilhámos.

Aos meus amigos que pacientemente me ouviram e se interessaram ao longo destes dois anos.

Por fim, à Escola Superior Gallaecia por me proporcionar a oportunidade de explorar este tema em particular nesta dissertação de mestrado integrado em Arquitectura e Urbanismo.

Foreword

This dissertation is mainly directed towards the continuous search of new methodologies that enable the perception of urban space through social behaviour's scope, while making use of the potential of computer's high capabilities of calculation. Although this architecture research is meant to be a scientific one, is also highly experimental and deeply linked with fields of knowledge that at early stages were far from being properly acknowledged namely: social behaviour, computer programing and perception of space, thus becoming long process of continuous learning of these specific areas.

The notion that there was another dimension beyond the three plus one that we're used to deal with in our lives, was the enduring question that fed the constant search for a way to achieve it and materialize it. At a starting point, the main goal was to try to recreate a computer model to be able to extract and materialize this intangible "without a name" feature that supposedly lived inside urban space.

One of the reasons that effect the decision of writing this in my non-native language was due to the fact that all the written information about these topic was, almost exclusively, in the English language. The second reason was that the intention, from the beginning, to take this research further beyond this dissertation was always unquestionable, thus the English language would provide a better way to expand a possible future audience interested in this subject. This also leads to the third reason, which was a way to improve my English writing capabilities (which it did).

Lastly this dissertation is aimed at those who also seek this feature and want to acknowledge it and materialize it.

Abstract

The focus of this research is to present a **method** to understand **complexity** within our **cities** from the standpoint of the **Social Sciences** and **Urban Architecture** itself while making use of Agent-Based Modelling and Swarm Intelligence **computational methods** based on simple **theoretical models** of **self-organization to** allow the exploration of, otherwise, **imperceptible qualities** within urban space as a complex network.

These methods provide an overview on how the human element in the urban environment behaves in a dynamic way and how the city operates from its core individual entities as an **emergent complex system** built upon **architectural elements** and **society rules** as well as basic human rules of motion.

Due to its **non-linear** approach to these **non-linear** concepts and principles, the structure of this dissertation takes several moments of exploration throughout its course along with the development of a **computer program** that keeps up with these constantly evolving principles and immaterial qualities.

What is then materialized through the adaptive application are **complex abstract shapes** that are perceived and **translated** into context specific **non-tangible spatial features**. Is a space open or closed? How close or how open? Maybe both? Or time-based features like flow/flux and tendency paths, like positive, negative, constrained, loose, etc. These are some of the expected translations that will take place in the closing statement of this dissertation

How these sets of features can be perceived translated and thought upon on a context-specific urban space are the main motivation of this work in a physical and in an abstract manner.

Both objectives of this research were fulfilled by providing a constantly developing tool that uses the potential of agent based and swarming models for simulation and urban space analysis and perceive emergent proprieties within urban and social space that are complex by nature as well as to review its potential in the integration with architecture's cultural techniques and practices.

Keywords: Complex Systems, Urban Architecture, Self-organization, Computational methods, Complex Networks, Swarm Intelligence, Social Complexity

Resumo

O foco desta invetigação é apresantar um método para compeender complexidade nas nossas cidades sob o ponto de vista das Ciências Sociais e da Arquitectura Urbana fazendo uso dos métodos computacionais, Agent-Based Modelling e Swarm Intelligence baseados em modelos teóricos simples de auto-organização para procionar a exploração de características, de outra forma imperceptíveis dentro do espaco urbano como uma rede complexa.

Estes métodos debruçam-se sobre como o elemento urbano se comporta de forma dinâmica no ambiente urbano, e em como a cidade opera desde as mais básicas entidades individuais, como um sistema complexo construido com base em elementos arquitectónios e regras sociais como também nas regras básicas de locomoção humana .

Devido à sua abordagem não linear a estes processos também não lineares, a estrutura desta dissertação assume momentos de exploração ao longo do seu desenvolvimento em paralelo com o desenvolvimentos de um programa de computador que acompanha estes princícios e características imateriais, em constante evolução.

O que é posteriormente materializado através da applicação são formas complexas abstractas que são percepcionadas e traduzidas em características espaciais não tangívies mediante contextos específicos. É um espaço aberto ou fechado? Quão fechado ou aberto? Talvez as duas? Ou características temporais como fluxo e percursos de tendência positivos, negativos, apertados, soltos, etc. Estas são algumas das traduções que farão parte da interpretação final desta dissertação.

Como estes conjuntos de características podem ser percepcionadas traduzidas e pensadas em contextos urbanos específicos são a principal motivação deste trabalho de uma forma física e abstracta.

Ambos os objetivos desta pesquisa foram cumpridos , fornecendo uma ferramenta em constante desenvolvimento que utiliza o potencial dos modelos de agent based e swarming para simulação e análise do espaço urbano e perceber propriedades emergentes no espaço urbano e social que são complexas por natureza , bem como para avaliar o seu potencial na integração com as técnicas e práticas culturais da arquitetura .

Keywords: Complex Systems, Urban Architecture, Self-organization, Computational methods, Complex Networks, Swarm Intelligence, Social Complexity

1 Introduction

This first chapter sums up the problem context and basic structure of this dissertation as well as some of the elementary principles and objectives that will allow for a systemic understanding of the process behind the development of computational methods to recognise emergent phenomena within urban and social space.

1.1 Problem Definition

The main substance of this research is to determine and explore computational methods to deal with complexity phenomena that happens inside urban and social space. These phenomena emerge from human society and its interaction with and within urban environments. This is achieved by gathering collective virtual data and developing logically consistent theoretical models to interpret patterns within that data generated through the computational models. (Social Complexity Academy, 2016)

1.1.1 Research Context

As society and technology evolves, man is led to adapt to the environment he is in, as the big webs that are the social networks connect the social system. It is essential now more than ever, to rethink the way architecture is conceived and perceived as all these new evolving concepts continuously shift to reconsider how space can relate to these new notions and how does it adjust to this new way of living within urban space. “For the current avant-garde the concept of space has lost its lustre. Contemporary avant-garde architecture is operating with new concepts, logics and methods that are no longer captured by the idea of space-making.” (Schumacher, 2010, p. 411). As Shumacher stated, this leads to a “new way” of approaching Architecture. Not one that focuses only on building and rebuilding the world in a physical way, yet one that emlbraces technology as part of the process of handling the complexity that emerges from within urban and social space. The need for new tools to embrace the need to conceptualize our perception of space must be the present-day architect´s concern, not only to use these tools to simplify the process of designing architecture, but to turn the whole process of conception dynamic by assembling new tools and really reformulate the spectrum of methodologies that nowadays are taken for granted. “And not allow pragmatic concerns to fall back on old models (Schumacher, 2010, p. 256). The experimental way of structuring processes through computational methods towards the approach of the urban design from its inception to its production is at the foundation of this research.

What is intended to be analysed in this research are urban spatial features and qualities, not only physic wise but also and most importantly through the social behaviour scope. Most of the methodologies used nowadays to analyse urban social space are one of two ways, mostly they are done through sociology and anthropology scope which are heavily based on quantitative theoretical generalization analysis. These provide consistent quantitative data related to the population that

is being studied but lack of the potential to combine that data to get a perception of the whole system. The second one is accomplished by utilizing methodologies through computational tools that nowadays are increasingly more common. Agent-Based Modelling (ABM) techniques sum up the ability of cross-referencing a population and their global social behaviour but also having into account the individual element of this system. On the top of this dissertation lies the intention to merge two different approaches, the social behaviour notions that come from sociology with Agent-Based-Modelling and Swarm Intelligence (SI) techniques through an urban architecture perspective. By doing this it is expected to harvest not only the arithmetical sum of results of these two fields, but a different perception of how cities behave, specifically how society behaves within it, with humanity as the main actor.

The context of this dissertation is to perceive how man as individual and as an element in a complex system that is urban space, lives and at the same time builds the city and is built by the city in a two-way relationship. How the set of his actions can be condensed to a set of behaviours some defined by society in the form of rules others by the external factors like physical elements and lastly by other individuals like himself lined by the same set of rules.

Like Alexander's reference to the concept of "the quality without a name", this is the one abstract feature that is intended to be extracted through the study of complex systems and then materialized formally into complex abstract shapes that can be, in a latter instance, translated into tangible and non-tangible features of social urban spaces.

> The fact that this quality cannot be named does not mean that it is vague or imprecise. It is impossible to name because it is unerringly precise. Words fail to capture it because it is much more precise than any word. (Alexander, 1977, p. 29)

How these group of factors can be translated to a feature that can be extracted and then observed and translated and how this feature can be used to get to know how cities operate not only in a physical way but in a non-tangible abstract way is the core subject of this research.

1.1.2 Preliminary Questions

The next series of questions were fundamental to the development of this dissertation as they constantly provide a focus and bring objectivity throughout the nonlinear process of this research.

What is the subject of this research? As stated before, the focus of this research is to understand the phenomena that underlies within human society and its reciprocal relations with urban spaces. This creates a series of questions. The first one is, in a way, already answered above and covers the main matter within the actual definition of the problem. The main subject is conducted towards

the **complexity within urban and social space** which will be defined as a primary subject and the main motivation behind this research.

How can it be achieved? It was important to determine which methodologies were appropriate to deal with these types of systems. As stated by Vehlken (2014, p. 11), "Agent-Based Modelling (ABM) and Swarm intelligence (SI) help configure environments that are increasingly confronted with the task of organizing highly engineered and interconnected systems, as well as that of modelling complex correlations."

ABM and SI methodologies are implemented through software applications in order to work with complex virtual systems inside a simulated space and these results translate into human behaviour. Using the algorithmic logics of ABM and SI and relate them to urban space and social behaviour provides the potential to actually recreate interactive processes that happen within the city inside a computational virtual space. One common question that emerges from these type of approaches is the real complexity of human behaviour, the power of choice, free-will per se. Although these concepts are assumed to be complex enough, it is important to realize that these methodologies are applied with reductionism, on the simulations of what exactly is being simulated. They don't simulate actual people, they simulate a model of people, and specifically they simulate the general motion of groups of people based on simple rules of motion.

> Software of this nature extends the possibilities of handling and optimising the complex interaction of various input variables for building processes. It integrates the levels of individual movements of particles (simulated humans, traffic flows, winds, etc.) At the mesoscale of single buildings and at the global level of urbanscapes. (Vehlken, 2014, p. 11)

This statement by Vehlken enlightens the idea of control of several interactions at the micro level (individual) and integrate them in a single system that reflects into readings at the meso-scale in this case, the scale of the building and at the macro scale in case of the global urban space. This is clearly a bottom-up approach as all the actions emerge from within the system rather than from a global plan. In another words it's like "getting a bunch of small cheap dumb things to do the same job as an expensive smart thing" (Corner & Lamont, 2004, p. 10)

It removes the ability for the user to control the system as a simple sum of its parts. This lack of control provides a certain filter between what may or may not have been a collection of biased decisions on the whole without having the knowledge at all times of the system that is being applied to.

ABM and SI act as problem solvers for this particular issue of handling complexity inside a simulation although the results it produces are a problem themselves as the data that is extracted is also complex in its nature. The third question is the answer to this problem.

How can this graphic information can be interpreted? The end result of these type of simulation is mostly graphical, and still difficult to interpret in a first instance. Again, Alexander's "quality without a name", or rather the extracted features remain intangible and must be processed through a "filter" in order to understand the meaning of each outcome. Latter in chapter 5 a more detailed approach to this problem will take place to help fulfil this interpretation.

Basic strategy concepts for a simulation:

Find a subject or a theme

Gather all information on that particular theme

Implement the functions

Fine tune the variables

Extract and interpret results

Why should architecture theory embrace these types of methodologies? According to Vehlken (2014), these procedures can be integrated in a in a contemporary cultural theory called Cultural Techniques which "originates from an agricultural discourse from about 1900 that has been revived for cultural analysis by a number of German cultural theorists in order to put emphasis on the dimension of techné inherent in cultural practices." (Thomas Macho, Sybille Krämer, Horst Bredekamp, Hartmut Böhme, and more recently Bernhard Siegert, Harun Maye and Erhard Schüttpelz, 2012, p. 12). They are comprehended of as actor-networks that involve humans, architectural objects, and the respective nexus between them. In these relation chains, not only do humans make use of technical things or design them according to predetermined techniques, but also technical objects locate human interactions inside the city and take agent-based-modelling into shaping their behaviour. This networks of functions and relations between actors, say, humans that complete certain task are catalogued into smaller specialized segments in order to complete a common task directly relates, in a way, to how society operates and consequently how the city performs as a whole. This leads to enquiry on why architecture can actually embrace these cultural techniques and integrate them in its own cultural techniques which, according to Dirk Baecker, in an interpretation that follows Niklas Luhmann's system theory. (Vehlken, 2014, p. 12) "considers the distinction between outside and inside as the fundamental cultural operation of architecture leading to the assumption that different cultural processes can be examined in order to contribute to specific architectural knowledge.

This is of significant relevance in the actual subject that take place in this research, the boundaries between man and intangible and physical space. How they work together as a complex system in a broader spectrum than simple linear action-consequence interactions, barely perceived using traditional models and techniques which also lies within, the distinction between outside and inside.

In a second instance, lies the relevance of the relation between time, space and man and the roles that the architectural grid patterns play in the control of this structure composed by these three fundamental elements, defined by Siegert, stated by Vehlken "technique that helps to structure and control space" (Vehlken, 2014, p. 13). The SI and ABM techniques are built upon a potentially unrestricted number of movement processes that not only define the emergence of boundaries between inside and outside during the simulation runs but also stores the data it produces for further analysis. This is of some relevance on extended urban analysis since, in these cases, several simulations are running on the top of the previously stored data providing a cumulative, timeline based, set of results.

> Their synthetic character (ABM and SI) is founded on an underlying algorithmic structure that defines neighbourhoods among all kinds of objects. In this case, space as such no longer has to be organised or constituted by a defined geometric grid, but self-generates out of the multiple local interactions of point clouds or particle swarms. (Vehlken, 2014, p. 5)

It's this kind of information that is intended to be extracted, a set of stored information that related to human locomotion and behaviour that can represent time-based features of space.

How can these results be translated into a real-world scenario? This last question is related to the need to create an overview on how these systems can be translated to the physical world of what is actually being simulated. This overview comes in form of a model, the actual model that is being used to apply the agent based methodology.

As a result of this definition one can describe them based on visual perception taking a level of abstraction to understand what is being dealt with in each simulation. This model can be thought as an analogy that bridges the virtual and the physical and to define what is being represented inside the simulation.

How are ABM and SI systems adapting to architecture procedures and how the architectural world uses them? And in what way? This is crucial in the actual simulations that take place in this research. The boundaries between man and intangible physical space. How they work together as a complex system in a much broader spectrum than simple linear action-consequence interactions, barely perceived using traditional models and techniques as they take several elements into account in real-time.

The synergy between society man space and time, is what architecture should be caring about. The abstract representations of more than the sum of these parts. This should not be a difficult role for the present day's architect as he's been doing this for centuries. The abstract representation of a

real object through a technique called drawing. The next quote by Patrik Shumacher bring this up referring a pertinent phrase made by Robin Evans.

> To operate with (more or less abstract) representations (drawings or digital information structures) has been architecture's predicament ever since its inception as a discipline distinguished from construction. As Robin Evans pointed out, architects do not build, they draw. Today Evans' dictum might be rephrased as follows: architects do not build, they script digital models. The central message remains the same: architects do not build. They manipulate representational models within a particular design medium. (Schumacher, 2010, p. 399)

Actually, it is relevant to point out that the greater difficulties of using computation to "draw" architectural concepts are the same as the ones when using traditional drawing, and is that, "The translation from model to building remains problematic." (Schumacher, 2010, p. 399)

In the next section some of these concepts will be analysed not only from the urban architecture's standpoint but also through a Sociology's scope.

What is the difference between these two methods (ABM and SI)? Agent-based modelling is a method that integrates a set of relatively simple mathematic functions and merges them into an algorithm, taking advantage of computation processes to calculate these functions individually and interactively combine them together. These functions share most of their variables and negotiate an end result base on an input set of variables.

On the other hand, Swarm Intelligence, uses the same logic but the algorithms it integrates are inspired by nature processes of self-organization and cooperation like flocking of birds, swarming of insects or schools of fish. The end result generally represents graphic similarities to animal movement.

In this research, these methods are used to simulate how people behave within a certain space, and although the mental process are apparently different, the basic rules of motion an interaction have similarities. These similarities embody the common ground in which this dissertation is supported.

1.1.3 Dealing with Complexity within Social Urban Environments

To understand general complexity, one must first understand what a complex system is and what principles it follows.

> A complex system is typically defined as a system that is "more than the sum of its parts." While the individual elements of the system may be incredibly simple and easily understood, the behavior of the system as a whole can be highly complex, intelligent, and difficult to predict. (Shiffman, 2012, p. 299)

As stated by Daniel Shiffman, a complex system is a system composed by smaller and simpler micro systems with "short-range relationships" that trade information forming a network where the end behaviour results in more than the elementary sum of its parts. It is important to notice that these simple units "operate in parallel" allowing the system as a whole to exhibit emergent phenomena. According to Colchester (2016), emergent phenomena is the end result of nonlinear interactions of many different actors.

In summary dealing with complexity its dealing with complex systems that produce highly unpredictable emergent phenomena.

What are the basic principles of complex systems? When talking about these principles, ABM and SI, are referred and the same thing with different approaches. There are three features that commonly a complex system follows, this also applies in ABM and SI simulations which are complex methods by nature.

The key principle of approach in this research is **non-linearity**. This means that a system, or rather, the smaller parts of a system, are greatly dependant on each other and sometimes a small increment on a specific value can turn the whole simulation to a whole different result. This is commonly known as the "butterfly effect" originally addressed by Edward Norton Lorenz in 1961 when running a simulation on weather prediction where a minor change in a variable resulted in a sunny prediction instead of a storm one.

In summary, a non-linear system is one that has no linear relationship between a change in initial conditions and a change in the end result and "A small change in initial conditions can have a massive effect on the outcome. Non-linear systems are a superset of chaotic systems." (Shiffman, 2012, p.300). This leads to the premise that complex systems deal with highly sensitive variables and specifically in this research, urban complex systems take a certain amount of effort to calibrate in order to get the intended result. In chapter 4 section 4.1, a deeper approach on the strategy to achieve a specific goal will take place.

1.1.4 Expected results

In a first instance it is expected to extract results from the developed application that can then be translated in to a real-world scenario. Immaterial features and concepts like simulated social behaviour, urban flux, tension, loosens and inside/outside relations within the urban environment

are the features that are meant to be extracted. It is important to notice that, complex systems are being dealt with, consequently, the results can be highly unexpected and unpredictable and as stated before and difficult to translate in some cases. There are though, some strategies that can be applied in order to obtain relevant and consistent results. These strategies will be addressed in the beginning of chapter 4 before the models are applied.

1.2 Objectives

- To define a set of computer models that enables the simulation of complexity within urban and social space through computational methods.

This is the core objective of this research. To define and to explore a toolset of computational methods in order to perceive emergent proprieties within urban and social space that are complex by nature. To formalize the non-linear interactions into emergent phenomena that are too complex to perceive simply by pure observation or by using traditional linear methods.

- To develop a computer application that allows for the implementation of the constructed models into context specific scenarios and translate its results.

This objective represents not only the development of the software itself but also the stage where the exploration of the potential of ABM and SI in the Architecture field takes place. The actual computer program development happens in parallel with the ABM ad SI theory across the whole research and it becomes an uninterrupted form of exploration. This also refers to the final stages of this research where a more practical, focused application of the model happens through the use of the continuously developing computer program and where the extraction and interpretation of results take place.

1.3 Research Methodology

The methodology adopted in the making of this dissertation has three main modes that will be explained in each section and can be directly translated into the chapters 3.1, 3.2 and 3.3.

In the development of this reserach, agent-based and swarming concepts as well as architecture principles are studies in parallel. At a starting point, agent-based concepts based on the Craig Reynolds “boids” models are integrated into a simple code. In a second instance architectural elements are implemented, like buildings, blocks and streets. The following phase was to integrate them in the sketch program and make the ABM elements recognize the architecture elements.

These methods are constantly combined with experimentation to fulfil a goal in understating architecture using computer simulation through experimentation processes.

The methodology adopted in the making of this dissertation has seven main stages that will be subdivided in each section and can be translated throughout the following steps.

The first stage of this research started with the initial question on how can computation methodologies be of use in the study of emergent architecture. The second stage in this research was to explore the possibilities of agent-based modelling methods and simulations that would allow to further analyse different features that happen within general urban environments through

the documental analysis technique. The first step taken at this stage was to find which available tools could be used for this task.

After some exploration on some programming environments, the most used and the one with most potential for this type of assignment was Processing which is an IDE Integrated Development Environment. This process required learning how this tool operates, and assume that it would complete the function that would be assigned to. Approximately half of the time of this research was focused on learning how to program in this JAVA based IDE while at the same time develop the tool that would lead to assembling one final application that would meet the needs of the problematic framework.

The process of conception of this dissertation, as referred before, is not linear and it goes back and forth. The schematic by Alessio Erioliin in the following images demonstrate the dissimilarity between linear and non-linear processes.

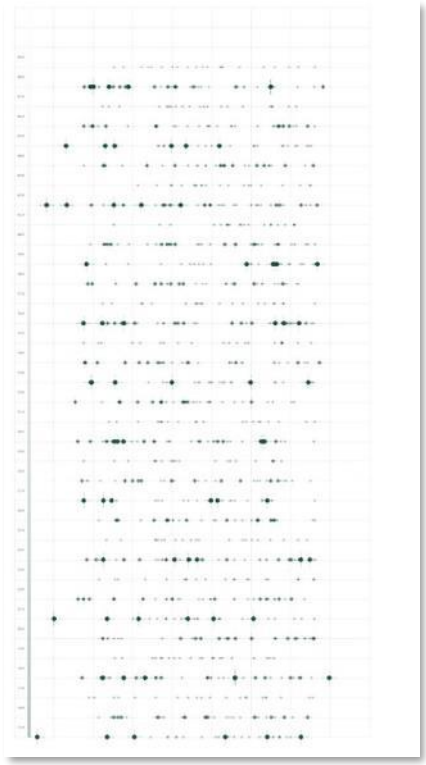


Figure 1 | Linear process

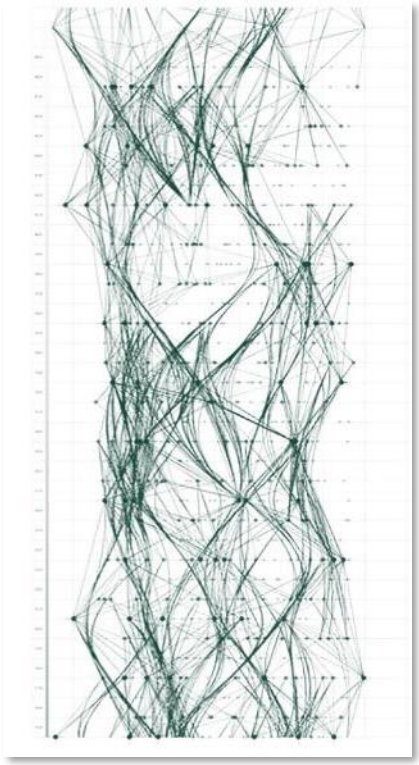


Figure 2 | Non Linear process

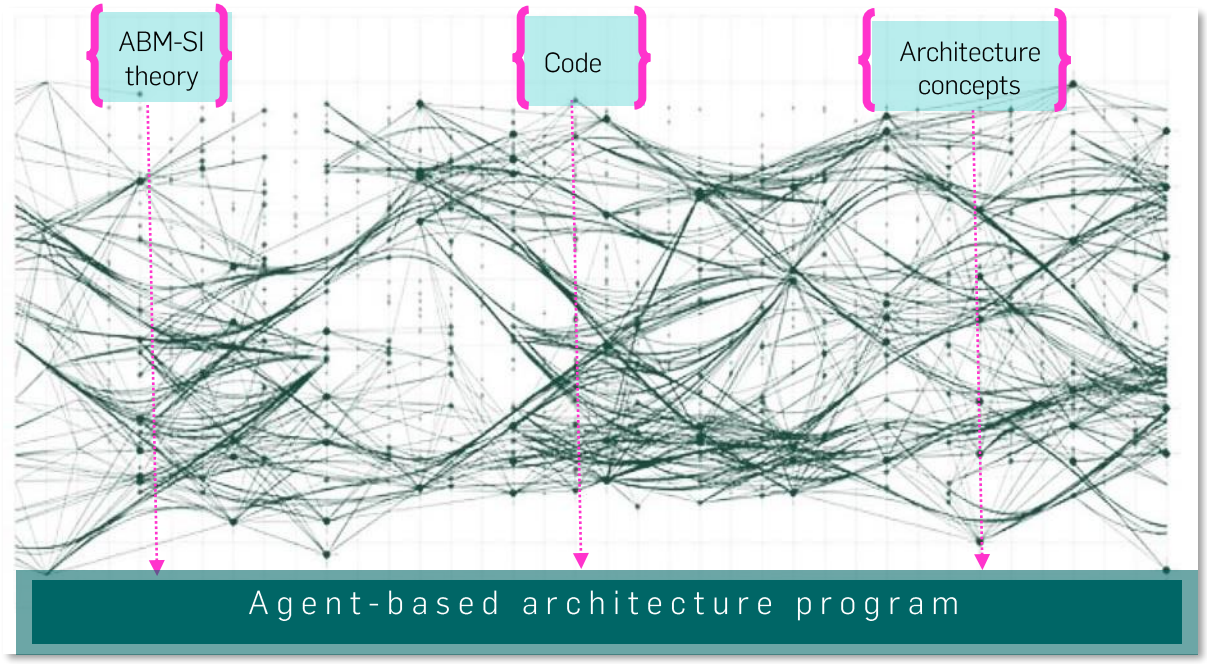


Figure 3 | Non Linear process with ABM, code and Architecture combined

Figure 3 represents a superficial idea of how this dissertation was developed, on the vertical axis, the process was direct and structured, on the horizontal axis, the nature of the subject obliged to an explorative non-linear approach where the computation models were interlaced with basic rules of motion and urban architectural concepts, resulting in a disordered yet structured methodology. As Stuart-Smith refers: "...looking at the design from the smallest element and the way that generates order at the macro level." (Snooks & Stuart-Smith, http://www.kokkugia.com/about, 2016)

The first stage in the research of this dissertation was to explore the possibilities of agent-based methods and simulations that would allow to further analyse different features that happen within general urban environments. The first step taken at this stage was to find which available tools could be used for this task. After some exploration on some programming environments, the most used and the one with most potential for this type of assignment was Processing which is an IDE Integrated Development Environment. This process required learning how this tool operates, and assume that it would complete the function that would be assigned to. Approximately half of the time of this investigation was focused on learning how to program in this JAVA based IDE while at the same time develop the tool that would lead to assembling one final application that would meet the needs of the problematic framework.

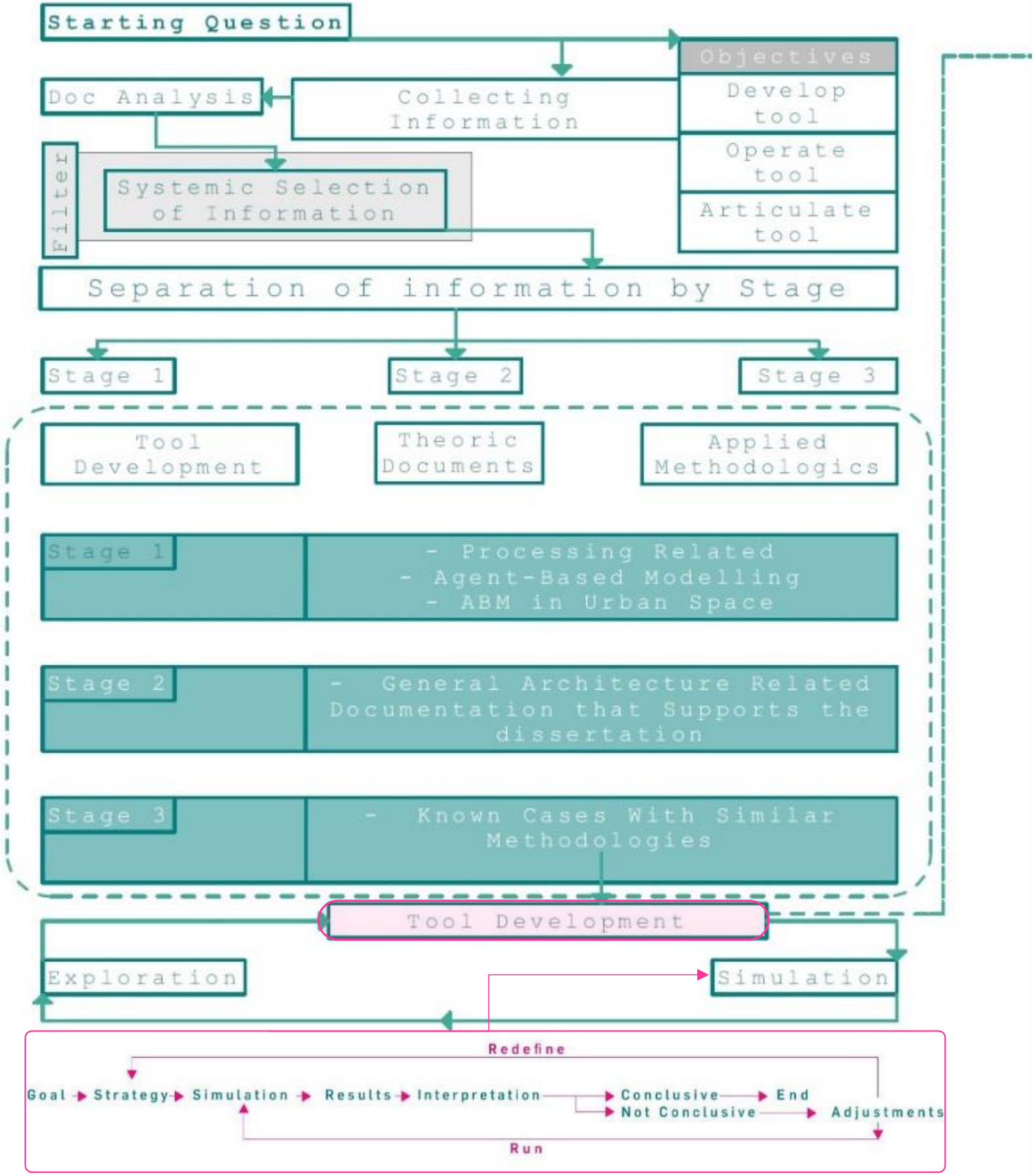


Figure 4 | Diagram of the stages and processes of development of this dissertation

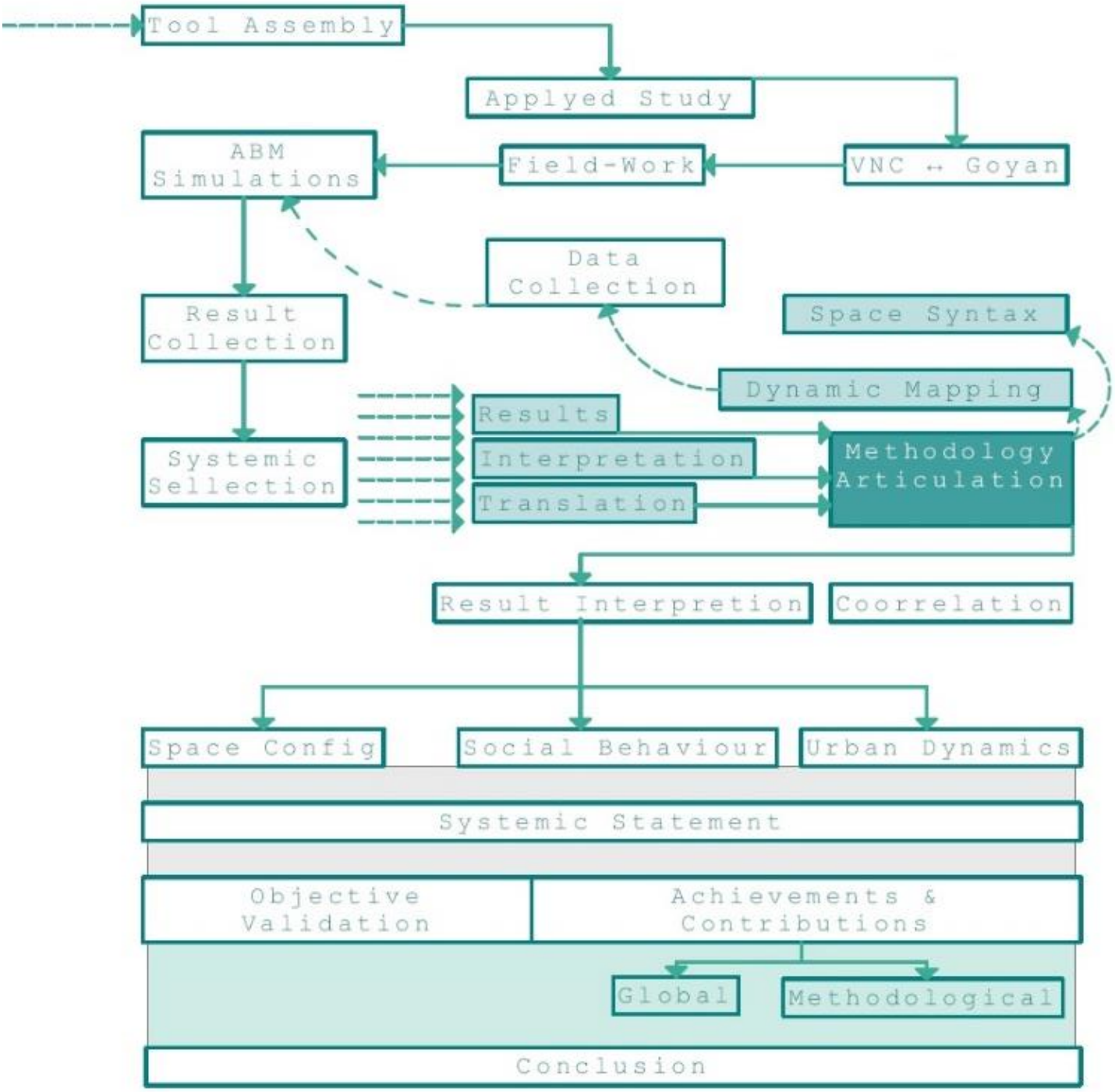


Figure 5 | Diagram of the stages and processes of development of this dissertation (part 2)

In the context of the research, a methodology was developed that combines different disciplinary approaches in a flexible operating structure, a non-linear development, adjusting agent based methods and swarming to context specific urban spaces and grounded by the specific authors of each matter. First the agent based models and rules of motions supported by the work Craig Reynolds reviewed by Daniel Shifman.

plane, the foundations for the relevance of these type of methodologies in architecture are settled and deeply driven by the work of, in and practical manner by Roland Snooks and Robert Stuart Smith at kokkugia, secondly, and in more theoretical approach by Sebastian Vehlken.

The methodology is based on qualitative and quantitative methods, directing the focus on urban social behaviour not only from the physical locomotion based rules supported by Reynolds - namely agent behaviour rules, but also in a second layer of complexity that results on mixture of these rules inside a simulation to later extract results that are graphically consistent.

The importance of this combining or rules is stated by Vehlken in a 2014 paper on Computational Swarming: “SI and ABM help configure environments that are increasingly confronted with the task of organising highly engineered and interconnected systems” (Vehlken, 2014, p. 11).

This process constitutes the theoretical support of the study of “architecture as information flow” and it justifies the implicit agent-based and swarm behaviour methods to this methodology.

1.4 Structure Outline

This section summarizes the basic structure of this dissertation as well the reasons why it was structured that way and not another.

This dissertation is divided into five main chapters:

Chapter 1 – Introduction

Introductory chapter that contextualizes the research as well as the preliminary questions objectives, research methodologies and result projections.

Chapter 2 – Theoretical Background

An overview of the state of the art related to this subject where all of the concepts related to ABM and SI, will be enlightened as well as some basic concepts about programming and how they are being used in the field of architecture.

Chapter 3 – Model Assembly

This is where the development of the computer program takes place and the main functions are explained. This chapter is divided into 2 sections that took place in parallel. This happens in order to maintain coherence between the ABM and SI theory with the developed program.

Chapter 4 – Model Application

Subdivided into 3 groups, (unstructured, structured and integrated simulations), this chapter represent chronologically the development of the simulation program. Starts by the exploration phase of the unstructured simulations, followed by a more structured approach to achieve a final goal of structuring context specific simulations within real-world urban environments in the latter section.

Chapter 5 – Result Correlation

In this chapter, the translation and interpretation of the extracted results, and the validation of the concepts referred in chapter 2 takes place.

Chapter 6 – Conclusion

What was achieved, what was the contribution, did it fulfil the objectives?

What is the potential development in a future work?

What are the limitations?

2 Theoretical Background

In this chapter a summary of the knowledge related to several relevant matters addressed throughout the research will be condensed namely:

2.1 Complex Systems

Describes a series of processes in order to deal with these type of systems

2.2 Computational Methods

Refers to a series of computation methods that were selected for use in this research

2.3 Agent-Based Modelling and Swarm Intelligence

Provides with the knowledge of Agent-based Modelling techniques as well as the differentiation between these and Swarming Intelligence methods

2.4 Swarm Intelligence in Architecture

Defines the potential use of these methods through the point of view of architecture.

2.1 Complex Systems

As addressed in the previous chapter the main problem of this dissertation is dealing with complexity. In summary, what is proposed to be achieved is to develop a model through computational methods to simulate the complexity that happens within urban and social environment and extract graphic results through a set of computational methods, models and techniques.

If SI and ABM systems are considered to be innovative techniques that help to deal with complex architectural problems, there must be a separation between two types of self-organization principles, one looks at the dynamic generation of architectural forms in social insects, the other is occupied with the dynamic movement and adaptive capacities of flocks or swarms on the move (such as birds or fish). In terms of architectural design, they serve several functions: first, **they can be used to produce idea models**; that is, they can inspire new shapes for further design measures as an outcome of emergent processes. Such idea models would not take shape without the algorithmic logic of SI and ABM. Second, **they can be used to represent the dynamics of existing architectural spaces in a simulation system**, facilitating a play with parameters and a testing and evaluation of different scenarios. And third, novel fabrication techniques that translate virtual models into material fabric can be attached to these computational tools

> Due to its complexity, the evolution of cities is something that is difficult to predict and planning new developments for cities is therefore a difficult task. This complexity can be identified on two levels: on a micro level, it emerges from the multiple relations between the many components and actors in cities, whereas on a macro level it stems from the geographical, social and economic relations between cities. (Beirão, 2012, p. 13)

When talking about these principles, ABM and SI, are referred and the same thing with different approaches. There are three features that commonly a complex system follows, this also applies in ABM and SI simulations which are complex methods by nature.

One of them is non-linearity. This means that a system, or rather, the smaller parts of a system, are greatly dependant on each other and sometimes a small increment on a specific value can turn the whole simulation to a whole different result. This is commonly known as the “butterfly effect” originally addressed by Edward Norton Lorenz in 1961 when running a simulation on weather prediction where a minor change in a variable resulted in a sunny prediction instead of a storm one.

In summary, a non-linear system is one that has no linear relationship between a change in initial conditions and a change in the end result and “A small change in initial conditions can have a massive effect on the outcome. Non-linear systems are a superset of chaotic systems.” (Shiffman,

2012, p.300). This leads to the premise that complex systems deal with highly sensitive variables and specifically in this research, urban complex systems take a certain amount of effort to calibrate in order to get the intended result. In chapter 4 section 4.1, a deeper approach on the strategy to achieve a specific goal will take place.

2.2 Computational Methods

Agent-Based-Modelling (ABM) is at the core of this research, formal methods of emergence in complex systems set its background. Formal methods are new methodological advances based on new developments coming from mathematics and computer sciences. These new methodologies are now being applied through formal approaches within Architecture's and Urbanism' scope. From shape grammars, space syntax to cellular automata and several others, with their different approaches, different fields of application, different levels of abstraction and formalization.

> The intention is to promote the use of formal methods in the creation of new explicit languages for problem-solving in Architecture. These problems range from representation, to theory, critique, production, communication, etc., never ceasing to see Architecture and Urbanism as technological activities and well as artistic ones. ESAP (2015, November) formal methods in architecture, 3rd symposium recovered from http://archformalmethods.wixsite.com

What is suggested is that these methods are of great relevance to the architecture field as they provide methods to solve Architecture an Urbanism problem that wouldn't be solved otherwise.

2.2.1 Processing.org as a tool for a tool

One of the stages of this development process was using an IDE called Processing as a platform to accomplish all the concepts of agent-based modelling in an explicitly graphical manner. For this to be attained not only all the ABM Rules that proven to be useful had to be implemented from scratch into a single sketch of code but also to be represented in a graphically rich environment. At this point one can think of the code itself as a group of 4 main components. (One could divide the structure of the code in several other manners, divided by classes, by functions or by type, it all really depends on how the coding is implemented and it is a very personal process to achieve let alone the explicit explanation of the code itself). The first step was to study an accessible computer language that would deal with Agent Based Modelling (ABM) for studding emergent behaviour of cities at different levels and scales.

Figure 6 | Example of the development environment Processing.org IDE

Processing is a creative programming platform (IDE-Integrated Development Environment) supported by the Java language developed by Ben Fry and Casey Reas, which assumes the purpose of combining the different programming (and increasingly diverse) areas of digital arts, through structuring of visual media applications and interactive. Having had originally educational purpose - as a tool focused on teaching the graphical component that language, open source condition - quickly encouraged the participation of growing communities, increasing its development in areas such as performing arts, kinetic arts, data visualization interactive real-time experimental architecture, among other fields of artistic creation and applied research.

Particularly in the architectural perspective, there is the in-depth research to the generative level design and implementation of limitations associated with "traditional" methods - where the Processing demonstrates its greatest potential, allowing the user (architect-developer) define specific dynamic applications that allow putting into practice the processing conditions and complex rules in creating certain architectural objects.

The role of Plethora-Project library

The Plethora library is an agent-based swarm behaviour library developed upon an academic open source project created by José Sanchez. It is an implementation of agent based rules that gathers the main components used when dealing with the study of emergent behaviours. This library is implemented inside processing in order to easily operate the agent-based rules used and improve the accessibility into the simulations. http://www.plethora-project.com/

This library and its functions were used throughout the development of the applications used in this dissertation.

2.3 Agent-based Modelling and Swarm Intelligence

What lies at the core of these exploration is a technique that has been used for decades in several fields to study complex systems, from weather forecasts to medical purposes, but it's under the social sciences viewpoint that this dissertation is supported. "You write a computer program so that the computer, when it runs, does similar things to what you are simulating. In this case we are trying to simulate people interacting in a social way." Edmonds (2012) interview by Tiago Gomes

This research focuses on developing a computational tool to allow the use of Agent-Based Modelling techniques in the Architectural field as these are now emerging in more than a few vanguard Architecture studios with several different approaches. The main issue is that for each project there has to be a specific code that has to be written accordingly, so, the user has to program a different application for each project. One of the goals when creating this application was to give the user the ability to apply these type of techniques and interlace them with architectural elements and processes in an intuitive way so they can be applied to any context-specific project. It is although important to understand how each component method and function works in order to take the most out of each objective in each simulation. Reynolds' approach and Boids simulated flocking. This simulation was the foundation of the short film *Breaking the Ice* in 1987 that can be seen on the images below.

An agent is the smallest but most important element of ABM. It is an abstract digital entity that moves in a certain space and time following a certain set of rules that interact with their environment. These rules are interconnected with another agents (neighbours) as well as with the environment it's embedded in. "The term autonomous agent generally refers to an entity that makes its own choices about how to act in its environment without any influence from a leader or global plan. (Shiffman, 2012, p. 260) In order to betvter understand the concept of what an agent is and how it works, a closer approach to the work of Craig Reynolds in the 1980's will take place.

This study was conducted in a non-linear workflow as was the development of the tool itself. This has been done since the beginning of the research and really focused on understanding how a simulation called boids simulation from Reynolds in 1986 actually worked but most importantly how these methods could be useful in the simulations through a social urban architect' standpoint.

> Much of my work involves writing software to simulate various types of human and animal behavior. These programs control the actions of autonomous characters in virtual worlds. I started by simulating bird flocks and related group behaviors. (Reynolds, 1996, p. 7)

Although ABM was a great basis for this work, it was important to acknowledge what has been done in the field of architecture and how it was accomplished. To understand the real potential of these methods and their limitations was key in the development of this work. For that, there are a lot of great examples nowadays such as the extensive work developed by Roland Snooks and Robert Stuart-Smith in kokkugia that are mostly experimental but that contribute to the knowledge of understanding the actual use of these kind of methods within urban architecture.

Figure 7 | Example of the Swarm Urbanism approach in Melbourne | Roland Snooks + Robert Stuart-Smith

Kokkugia is an architecture research studio that focuses on developing methodologies to explore collaborative generative design that emerge from complex systems and self-organising behaviours, of social and material systems. The studio is led by Roland Snooks and Robert Stuart-Smith and are based in Melbourne and London. Kokkugia's approach to architecture is deeply speculative and demonstrates a strong intention on non-linear architecture. "One that emerges from the operation of complex systems and questions the established hierarchies that operate within architecture" (Kokkugia, 2016) recovered from http://kokkugia.com/about

Figure 8 | Form exploration of a pedestrian bridge using ABM and SI methods | RMIT HIGHWAY - RMIT | 2013 - Roland Snooks

Their approach involves encoding simple architectural decisions within a distributed system of autonomous computational entities, or agents. It is the interaction of these agents and their local decisions that self-organizes design intent, giving rise to a form of collective intelligence and emergent (Snooks & Stuart-Smith, http://www.kokkugia.com/about, 2016). The work of kokkugia in general, and specially the one described above in particular, added a huge contribution to these type of methodologies nowadays and how they can actually be implemented in real-world situations.

2.3.1 Autonomous Agents

An agent is always subjected to a set of rules and behaviours as described later in this Chapter and they will be addressed accordingly to those chapters' principles. In this case, Dynamic agents are sets that obey to a maximum speed that is bigger than 0 as opposed to a static agent like an attractor agent which in this case have a "Max Speed" of 0. "Max Speed" described later in this chapter.)

The next set of values defines the behaviours of the "Agent001" group. As it will be clarified later in this chapter, when "Reset Values" is activated or at the beginning of a simulation, each value assumes an initial default value that works for most situations and it's a good starting point as they follow the standard radius hierarchy described in chapter 1 (section 1.4.1.2.1.4). This structure follows back to the Craig Reynold boids rules and how they are implemented.

First comes cohesion, and cohesion radius specifically. "CohesionRadius" is generally the biggest radius of the three major rules in the case of dynamic agents. The reason can be perceived if one goes back to the origin of these rules, in the Reynolds simulation "Boids". There is a tendency for the flocks of birds (birds + droids = boids) to be close to each other with a certain safe distance. So, when an agent is too close to another but within the safe distance it aligns itself with its group making small adjustments in speed and direction in order to go back into a balanced movement. Following this logic is understood that "AlignmentRadius" is smaller than the "CohesionRadius".

Then comes the separation radius, when the distance between the agent and its neighbour is smaller than the defined "SeparationRadius" a force triggers, perpendicular to the tangent defined by the agent and the circle.

Generically speaking one can assume that the: CohesionR > AligmentR > SeparationR logic will succeed in most situations but this is not always the case for every simulation.

As described before in chapter 1 (section 1.4.1.2.1.4), each simulation has its own strategy and rules that can be tweaked for a specific run. Although this logic will apply in most cases, it might radically differ according to the situation, to a point where this exact logic can be inverted to get a specific result. Some examples of this reversed logic can be seen later on chapter 3 section 1 "Results" in

the tower example where agents in a “ConstrainedSpace” and where values of separation are much higher than cohesion.

As referenced before, an agent is an abstract digital entity that moves in a certain space and time following a certain set of rules that interact with their environment. These rules are interconnected with another agents (neighbours) as well as with the environment it’s embedded in. In order to better understand the concept of what is an agent and how it works, a closer and hypothetical approach to this entity will take place.

An agent is a point in a three dimensional space, not static point but a dynamic one. So it’s a point that moves in three dimensional space. The next step is to question: where/why/when/how does the point move?

One can start by imagining a point traveling endlessly throughout space in a straight direction, although it’s not an autonomous agent yet, it’s a start.

For an agent - or a set of agents - or any dynamic geometrical shape - to exist in this environment it must be created in a certain moment in time, at a certain location in space with a certain initial velocity. At the time = 00:00:00 an initial set of 100 agents at the initial location = (0, 0, 0) with an initial velocity of 1 (in a random direction) will be created in our constrained space. (Box (600,600,600)). In the real world, we can’t have the same object in the same location at the same time but in the “agent world” this condition can happen. So, one hundred points are now at the given location with a certain velocity. As time moves on they start to spread in a random direction.

The most intricate thing about agents it’s how they behave with each other and the environment. For this to happen one must create a set of rules for them to interact. Now a rule to the behaviour will be added. The name of one of the rules is called separation and it will bring some awareness to the set of agents. As the name implies, this rule’s purpose will be to provide the agents with the ability to separate themselves from each other according to two basic variables, radius and scale. So, if an agent is at a certain minimum distance from each other (radius) the separation function will kick in with a certain amount of force (scale).

Although separation plays an important role on swarming simulations, other types of rules must be implemented so that agents behave in a more natural way. These rules will influence drastically the global reaction of the emergent result. They represent the DNA of a specific set of agents thus must be fine-tuned carefully for a specific situation. Further in this research, these concepts and what they represent will be explained individually as well as how they work and how they can be used to perform specific tasks. The rules for the simulation can be divided in to two main levels, Low Level Rules (LLR) and High Level Rules (HLR). These simple rules will be within the group of LLR as they are at the core of a specific set of agents.

The LLR are the simplest rules and the ones that allow for the agents to react among themselves, and the HLR being all other rules that either will allow for the agents to react to external

components or to perform specific tasks. The amount of these high level rules can be endless since they are defined by the user and might become more and more complex within each iteration.

An autonomous agent must follow a specific set of rules that will describe his movement. These rules will define the primary behaviours of our group of agents and must be defined and controlled by the user. There are 3 basic rules for the simplest form of an agent to be implemented. These 3 rules where implemented by Craig Reynolds in 1986 on a simulation called boids. - In the late 1980s, computer scientist Craig Reynolds developed algorithmic steering behaviours for animated characters. These behaviours allowed individual elements to navigate their digital environments in a “lifelike” manner with strategies for fleeing, wandering, arriving, pursuing, evading, etc. Used in the case of a single autonomous agent, these behaviours are fairly simple to understand and implement. In addition, by building a system of multiple characters that steer themselves according to simple, locally based rules, surprisingly high levels of complexity emerge. The most famous example is Reynolds’s “boids” model for “flocking/swarming” behaviour (Shiffman, 2012, p. 263).

As it was assumed previously, an agent is an abstract entity that conforms to a set of rules. According to Reynolds these rules are commonly known as steering behaviours. These are the basic steering behaviours used in this graphic user interface.

2.3.1.1 Steering behaviours

According to Reynolds (1996) Specific steering behaviours like Cohesion Alignment and Separation assume that locomotion is implemented to a single agent and applied by a steering force vector. Therefore, the steering is described by thrust, braking, steering, seek steering, flee steering, desired velocity, seek target, current velocity seek path, flee path, desired velocity of the geometric calculation of a vector representing a desired steering force (...) generally the magnitude of these steering vectors is irrelevant, since they will typically be clipped to max_force by the vehicle model (...)

Cohesion

This steering behaviour gives an agent the ability to bind with other nearby characters. See Figure 4. Steering for cohesion can be calculated by finding all agents in the local neighbourhood calculating the "average position" (or "centre of gravity") of the nearby agent groups. The steering force can be applied in the direction of that "average position" (subtracting our singular agent position from the average position, as in the original boids model), or it can be used as the target for seek steering behaviour.

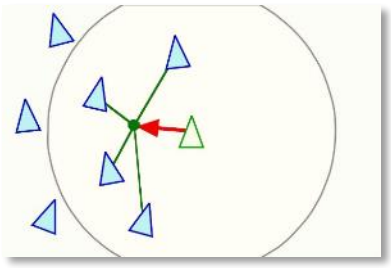


Figure 9 | Steering behaviours by Craig Reynolds (Cohesion)

Alignment

This steering behaviour gives an agent the ability to align itself with (head in the same direction and speed as) other nearby characters, as shown in Figure 8. Steering for alignment can be calculated by finding all agents in the local neighbourhood, averaging together the velocity of the nearby agents. This average is the "desired velocity," and so the steering vector is the difference between the average and the single agent's current velocity. This steering will tend to turn our character so it is aligned with its neighbours.

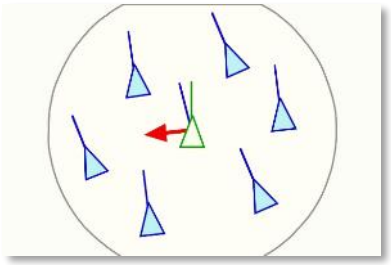


Figure 10 | Steering behaviours by Craig Reynolds (Alignment)

Separation

Steering behaviour gives an agent the ability to maintain a certain separation distance from others nearby. This can be used to prevent characters from crowding together. To compute steering for separation, first a search is made to find other characters within the specified neighbourhood (same process as described above for alignment). This might be an exhaustive search of all characters in the simulated world, or might use some sort of spatial partitioning or caching scheme to limit the search to local characters. For each nearby character, a repulsive force is computed by subtracting the positions of our character and the nearby character, normalizing, and then applying a 1/r weighting. (That is, the position offset vector is scaled by 1/r 2.) Note that 1/r is just a setting that has worked well, not a fundamental value. These repulsive forces for each nearby character are summed together to produce the overall steering force (Reynolds, 1996, p. 7)

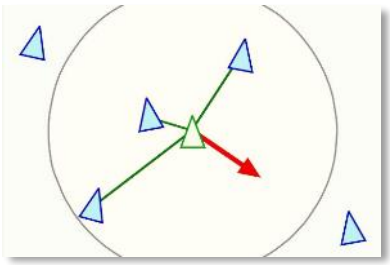


Figure 11 | Steering behaviours by Craig Reynolds (Separation)

Radius

These rules, also called behaviours, they have a general hierarchy with the specific order of radius sizes. They are implemented through steering forces that are calculated through vector operations. The cohesion radius is, in general bigger than the alignment radius which is also bigger than the separation which are described below. This can be perceived if one looks to how birds behave in a flock. Their first priority is to move as a whole so they are attracted to one another with a big radius.

The second one is, when they reach within a certain radius (noticeably smaller than the cohesion radius) they adjust their direction and velocity to adapt to the rest of the group. In a last instance, when the agents reach the separation radius, the separation force triggers in the opposite direction of the neighbour in range.

This is part of the implementation of the cohesion, alignment and separation rules explained above and it defines a circle with a specific radius controlled by the controller with same name around an agent (ex.: “CohesionRadius” or “AligmentRadius”). The radius of influence of an agent can vary according to the specific simulation.

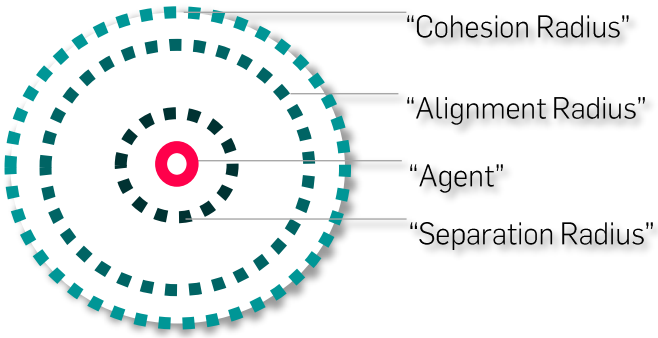


Figure 12 | Basic agent radius hierarchy

Wander

Wander is a type of random steering. This is singular type of behaviour relatively to the three referred previously since the whole implementation relies on a certain amount of randomness for the agents' locomotion. It takes the direction and speed of an active agent and randomizes a certain amount of the previous values in order to get a semi-random movement. "At one frame, the character may be turning up and to the right, and on the next frame, it will still be turning in almost the same direction" (Reynolds, 1996, p. 8). This function is useful when agents are spread through an unknown environment and that allows them to explore for elements in their surroundings.

Scale

At this point it is important to acknowledge the concept of scale in these techniques. Since its not physically based algorithm, scale is not simply defined by the "size" or "distance to the screen". So a simulation can have several scales. Not always a simulation has a geometric scale for example. Only when external geometry of a specific location is added the simulation gains this traditional concept of scale. This scale concept is latent along the whole application but never referred to as being a specific variable, in the end it represents all variables in one singular concept. This also means that scale is the actual one variable that is defined in each algorithm. It can make a simulation to go completely erratic to a simulation that is considered relevant.

The way this system works allows us to work in various scales and create different conditions for each simulation. It is always important to have in mind that each simulation is abstract and must be translated so it can have some interpretation. The same goes with its scale. One must fine-tune each variable for each test in order to get the best results.

Another example would be the force of an attractor, this force can vary from attractor to attractor but at the same time be subjected to a global clipping of that force, this variable is called max_Force and represents the maximum force being applied to a specific set of agents. This variable also represents scale. The scale that each force of each set unfolds from. This leads to think that different sets of agents can have different "force scales" depending on the set of variables each is subjected to. Cohesion scale, Separation scale, Alignment scale are other forms of scale inside each simulation.

2.3.2 Swarm Intelligence and Stigmergic Behaviour

In sequence of the previously referred concepts of agent rules by Reynolds which provided an overview on direct locomotion rules and the emergence of a general action-reaction behaviour, in the next section, a new level of intelligence will be introduced in order to understand how some of these concepts relate to society itself and how it works, as well how they can be of value in future simulations when applied to a constructed model.

At this point, one can say that the active application in question has three main engines that drive each simulation. Agents, attractors and geometry. This combination of elements allows for direct, one to one reactions that allow for the emergence of patterns of locomotion. These behaviours demonstrate simple principles and rules of locomotion based on variables of proximity. These simple rules as seen on the next chapter provide a more natural way for the agents to behave in a specific environment rather that simple reactions of "if this then do that" type of rules. Not only allows for more dynamic behaviour but also allow for "the accident" to happen, meaning that it adds a layer of semi-random decisions (wander function) to take place that allows for other emergent events to occur. These set of functions allow for each set to either react to themselves, to others of the same group or to other of different groups when they "see" elements around them. One issue that this type of model contains is the lack of memory in each individual agent. What the concept that will be introduced next represents is, not only the condition of "I see another agent" but on step further add the condition of "I saw another agent" going in a specific direction or also "I saw myself in the other day" choosing this route to work. Now "I" have a decision to make, "The previous route was better" or it "wasn't" and then react accordingly, allowing for an optimisation of the route taken. This concept can spread to a collective memory and can be a game changing method in this specific application since not only allows for a simulation to have more focused results but also adds the possibility for creating other functions that require this type of memory allowing for self-organized behaviour closer to what is seen on human behaviour and decision-making locomotion.

As pointed along this dissertation, a parallelism is being made to compare these type of simulations to real world scenarios in order tro extract relevant intangible information. What is going to be introduced, in a first instance when implementing this new function, might not be directly translated into general human behaviour, but instead, it will allow for the foundation of several other rules that will sustain these concepts of intelligence and memory.

Take for instance a simple simulation example with basic attractors; a set of agents is created and influenced (if in range), by these attractors. One of the questions that appeared while testing this concept was that, in a real world scenario, if a path was created by a pedestrian, the next time this pedestrian walks by the same area, the knowledge of this area would be sensed in a different way. This lead to think each simulation as a step of a larger and constantly evolving simulation.

> Stigmergy is a coordination mechanism observable in biological insect societies like ant colonies. It is based on the behaviour of these ants to create a dissipative field by spreading smelling substances into their environment. Ants permanently interact with this dissipative field, being guided by it to certain activities and changing it by spreading pheromones as a reflection of their activities. Stigmergy allows complex social behaviour to emerge from simple activities of single ants. It has surprisingly good properties in terms of flexibility and adaptability.
>
> (Valckenaers, Kollingbaum, Van Brussel, Bochmann, & Zamfirescu, 2016, pp. 1-2)

This concept of coordination mechanism between agents is called Stigmergy and can be explicitly recognized on nature, specifically on biological insect societies like ant colonies. The self-organized structure of these colonies functions in similar way, the decisions of each "ant" is based on a specific task and on the placement of smelling substances like signs or stigmas inside the environment by other "ants" that relate directly to the specific task at hand creating a synergy (Stigmergy = Stigma + Synergy). The cooperation of a specific community that works towards a common goal by interpretation of a set of signs. This set of signs is a "field" of, in case of this application, points that represent a group of attractors (negative or positive) that allow for the whole community to understand tendency paths and make a decision either to follow them or avoid them in order to optimize the network structure of the specific micro-society that is being simulated.

Later in chapter 4 section 4 this algorithm will be applied in order to explore how Stigmergy can be of relevance when used in simulations within urban space.

2.4 Swarm Intelligence in Architecture

It is intended to realise how context specific elements inside the city, influence every other element in a complex non-easily perceived process that is not simply revealed through simple first-instance observation or simply by looking at raw data studies analysis. Lastly to understand how these question can be of value when studying Urban Architecture.

This dissertation aims to develop a method for architecture exploration and creation having in mind the notion that time changes but mostly the synergy between man, space and time. Specifically, how man and space shifts throughout time "in a deeply connected way" (Schumacher, 2010, p. 387). But time is not the only variable that changes, all the other factors that must be considered when approaching these formal methods to a specific urban environment must also be part of this equation. This delicacy of dealing with systems within architecture that are too complex to resolve in a linear fashion are "far from being adequately addressed or explored theoretically, experimentally, or phenomenologically" (Kolarevic, 2015, p. 7) so the focus in this research is to provide an overview on how they can be used to structure a methodology.

Often when one creates an architectural object for a certain location, must always think of its surroundings and measure what impact these make in the created object as well as the impact this new object influences its surroundings. This process often describes a linear workflow (top down) that has a beginning and an end and follows a series of steps to achieve one goal, one final result that is the sum of all these steps ending in a final architectural object. One can think of it as an arithmetic sum (analysis + concept + structure = result). Although this has been standard procedure in architecture creation for centuries, it has some issues in adapting to the nowadays concepts of change in technology and connecting to the new networks that link our cities.

> "A two-way relationship could be established among the spaces, environment, and users: users or changes in the environment could affect the configuration of spaces and vice versa. The result is an architecture that self-adjusts, that continuously changes – an architecture that is adaptive, interactive, reflexive, responsive (...)" (Kolarevic, 2015, p. 7)

This leads one to think urban space as a whole focusing on all variables of a specific place at all times. This synergy has always been difficult to achieve since generally a linear process workflow is used. In this process of creating and investigating architecture some of these extensive variables are often either forgotten, overly thought, or either will have a relatively biased result being oversimplified by the architect. This gap between the ideal way of designing a specific space and actually doing it is not easy to fulfil, and should not be taken inaccurately. This research provides a tool based on these extensive variables one faces when first approaching space in a first instance in order to seal part of this gap.

In order to better understand the process of this method an analogy to the practises and tools one uses to create new tools will be arranged. For the subject of this analogy a simple object is going to be used. Our example object will be a shovel and to create it one needs to be familiar, in a first instance, of what a shovel is and what one needs it for as well as knowing that is made out of wood and metal and how can one work on these materials. In a second instance one needs to know how to work with the tools that allows us to carve and cut wood as well as forming and cutting metal. This analogy allows for a better understanding how a tool can be used to create a new tool for different purposes. In this research lies a similar procedure. One first needs to know what we want to achieve - what the shovel is and what it is needed for - and then decide on what tools to chose in order to build it. The latter being computer software and programing.

Following this procedure, one can now can say that a computer is used as the tool (and some of the tools inside it) to create a new tool (the actual shovel needed) to create and fulfil a gap in the design process and dynamic analysis process. The next step is to know how a computer works and how the tools inside it can be used to create a new tool. This is as far as this analogy goes.

> That's why we need the avant-garde: where there is methodical tolerance, where there are dry runs, experiments, and manifesto projects; and you can't expect to immediately compete with mainstream state-of-the-art. (Schumacher, 2010, p. 288)

For this tool to be functional, certain concepts must be enlightened in order to understand how a system like this can be assembled. Agent-based modelling is at the core of the software structure and it will allow for the user to simulate, explore and create space following a set of very sensitive and specific features. It is important in this kind of exploration to have in mind that all variables are relative to each other. Variables like space, attraction and repulsive forces and several others are not absolute, even time itself. So in a simulation, one addresses time as being relative to "our world" time by a specific factor, generally with a ratio bigger than 1, meaning that time in a simulation is almost without exception faster than "our time". On variables like space meaning, for example specific blocks that will influence our strategy on analysis and creation, the relativeness of these values that are applied as forces are very sensitive to one another, as well as to the other components around them. This makes the whole system very delicate and easily breakable so it must be treated very carefully in order to have suitable results. A graphic user interface will allow for the user to make these adjustments on the agent based system itself. These models lead to the concepts described in the next chapter that will provide a deeper understanding on how they can be implemented into an application.

In summary, this chapter described relevant theoretical models that support the development of the application build for this dissertation. These methods act as the main pillars for the simulations that took place in parallel with the application development

3 Model Assembly

The following sections (3.1, 3.2) can be read in parallel or independently as both are structured individually in a linear manner but each stage of the development was part of a non-linear process of exploration were the actual program was being incrementally written while being conceptually supported by the ABM and SI concepts interlaced in a two-way feedback process. This non-linear process is apparent throughout this research – particularly in this chapter and in chapter 5 - for some of the concepts are directly interlaced. These function links are summarized in the diagram below.

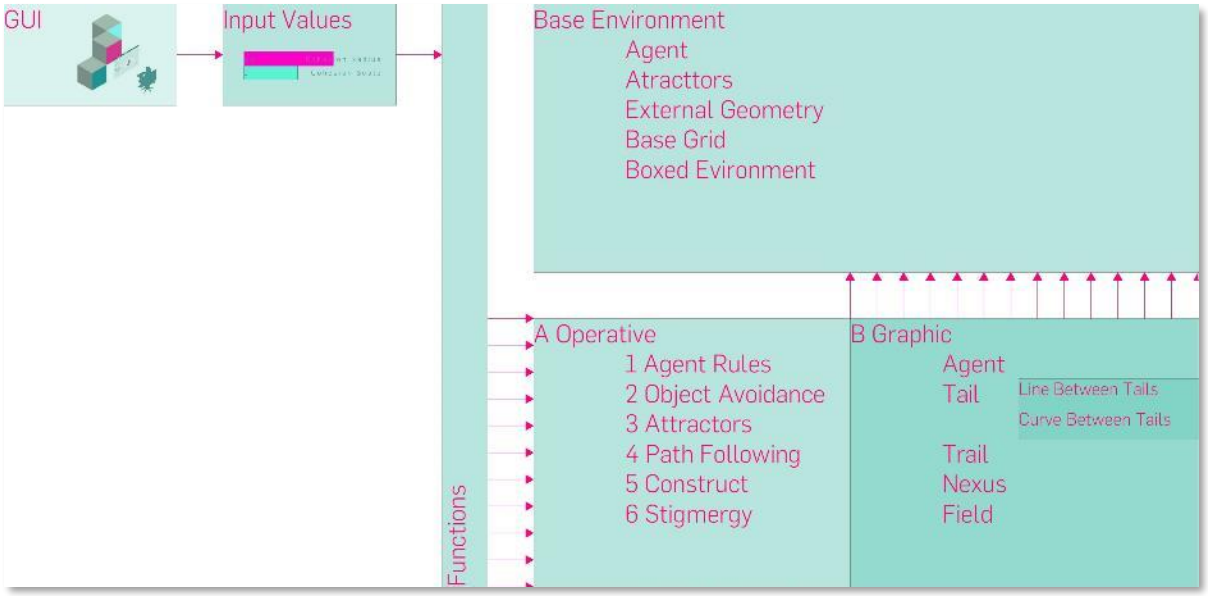


Figure 13 | Basic software structure of the developed application

The graphic user interface acts as a bridged between the user and the functions of the application. These functions can be either operative, like certain values for agent behaviour or graphical, meaning that they control what and how the information is being visualized.

3.1 Software Development

This section is intended to provide a linear description of the hierarchy that composes the structure of the actual object oriented program, from its inception (the process of learning a computer language) to its conception, that is, the last stages of the its development, along with the decisions that were taken throughout the complete process. One can see this structure as a layered one, for it is composed by parts that go on the top of each other in order to form a whole. This will clearly be observed in the Program Outline section.

On this stage of development, the IDE (Integrated Development Environment) Processing was used as a platform to accomplish all the concepts of agent-based modelling in an explicit graphical manner. For this to be attained, not only all the ABM Rules that proven to be useful had to be

implemented from scratch into a single sketch of code*, but also to be represented in a graphically rich environment. At this point one can think of the application as a group of 4 main components explicit in the chart above.

At this stage, one can say the first steps taken will influence deliberately the steps taken in the next stages, to achieve our purpose. One of the first steps needed after implementing the agent class was to control each rule with an interface. Controllers were implemented using the controlP5 processing library for the basic agent rules- This allowed for quick testing and have a visual feedback on how certain variables change the way agents behave, without having to reset the application and change the value inside the code.

In a second stage the development of the application takes a necessary turn and directs itself into a more user friendly interface. This was also a period when several core functions were implemented to what would later become the final goals to be achieved.

The Basic drawing of shapes buildings splines, among other functions, to make quick simulations, were implemented, while at the same time the system was tested. At this stage several results came out of this unconstrained experimentation. Although some of them are naturally more relevant than others, each one served a purpose on learning how the complex system behaves in a general way. The most relevant experimentations will be presented in chapter 4 (section 4.1) where a selection of the ones with most potential to architecture will be translated. One click creation and deletion of groups of agents that allowed the creation of a certain number of agents in a specific position at any time, was actually a turning point in the development of the simulation program in terms of user ease of use. This allowed for a more intuitive approach on learning how each variable and individual functions influence the general behaviour of the agents.

Implementation of imported external geometry also happens at this stage, which is an important one, for it will allow later in the following sections to use it to import specific architectural elements like grids, buildings, roads, etc. into the simulation environment.

In this final stage all functions were already implemented and the user interface was optimized in order to have a more effective access to any function at any given time. Although it's a condensed phase, some of the components inside are still being worked on and this is part of a future work on the development of this research.

3.1.1 Program Outline

The next series of infographics contain information about how the developed application is structured and will go in depth on how the most relevant functions operate. These functions are detailed in order to understand not only how they operate individually but also how they operate with each other as a whole system.

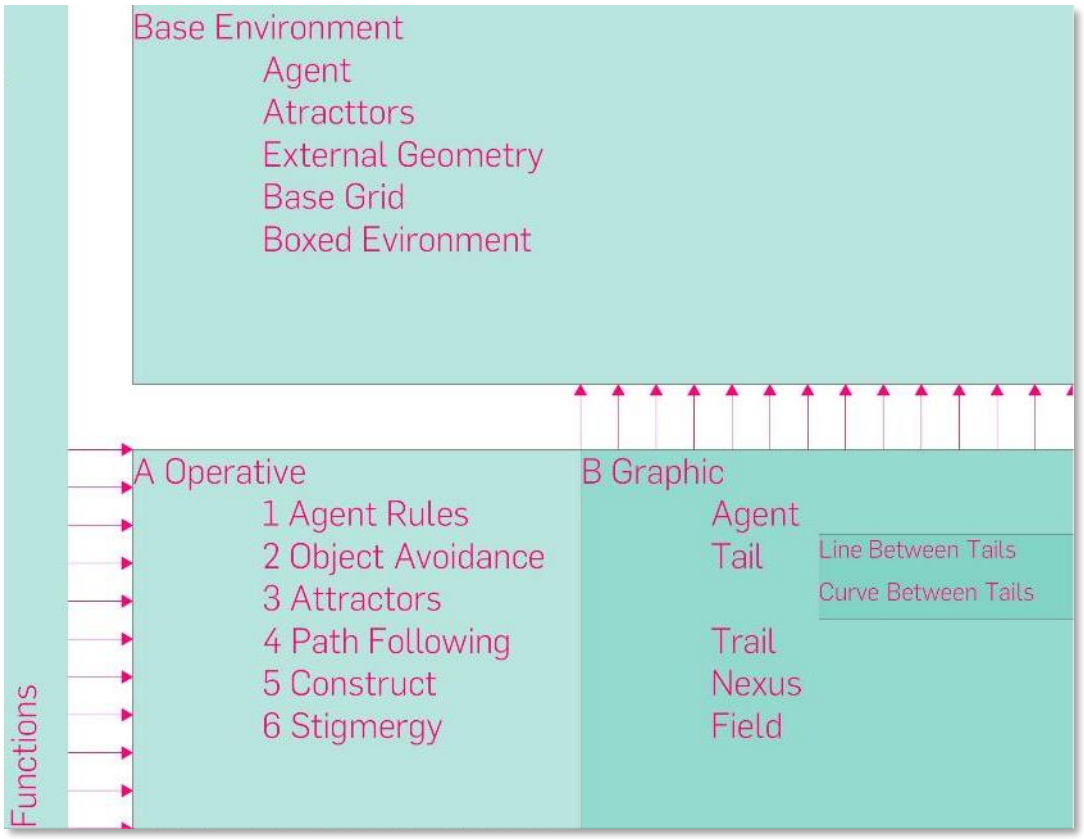


Figure 14 | Basic software structure of the developed application (2)

Agent rules are what define the global behaviour of agents. As referred previously on section 2.3.1.1. these are the basic rules of motion applied in the simulations. The next series of images represent the final stage of a set of simulations that took place in order to understand how each variable influences the general behaviour. These rules were tested in the same empty environment in order to visually perceive how different values would affect the behaviours of a set of agents. All the rules (cohesion, alignment and separation) were tested with the same scale. The variation of values in the next set was the cohesion radius, although several other sets of values were tested.

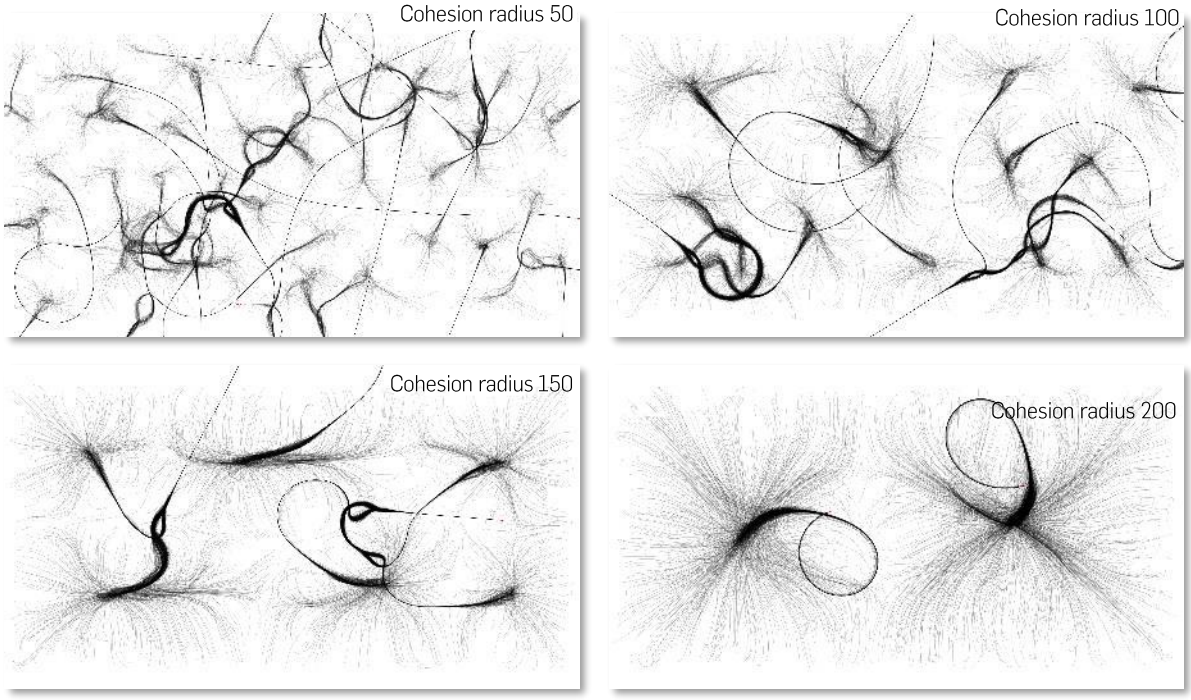


Figure 15 | Early results testing different sets of cohesion radius values

```
1 Agent Rules
  for (int i = agents.size()-1; i >= 0; i--) {
      Agent other = (Agent) agents.get(i);
  a) Cohesion
   if (this != other) {
    if (loc.distanceToSquared(other.loc) < cohNeighborDist*cohNeighborDist) {
      coh.addSelf(other.loc);
      count1++;
    }
   }
  b) Separation
  if (this != other) {
       float d = loc.distanceTo(other.loc);
       if (d < desiredSeparation) {
         Vec3D diff = loc.sub(other.loc);
         diff.normalizeTo(1.0f/d);
         sep.addSelf(diff);
         count2++;
       }
     }
  c) Alignment
      if (this != other) {
       if (loc.distanceToSquared(other.loc) < aliNeighborDist*aliNeighborDist) {
         ali.addSelf(other.vel);
         count3++;
       }
      }
  }
```

Meaning that for every agent inside this group calculate the following functions

If the agent number i (all agents) is in range excluding itself

Adds a force into the average location

If the agent number i (all agents) is in range excluding itself

Calculates a vector pointing away from the neighbour

Applies a force in the opposite direction

If the agent number i (all agents) is in range excluding itself

Figure 16 | Overview and code implementation of the agent rules.

Object avoidance is a function that allows for a set of agents to avoid imported geometry, in this case it represents buildings of a hypothetical location. The way it works is a set of points is imported and then connected by lines which are recognized by the application. These lines are then divided by points along each line with a certain distance between them. This distance is highly dependent on two concepts, scale and application performance. What often happens when this "wall density" has value that is too small or too big, some errors may occur. This variable has to be adjusted in every simulation, if the density is too small the agents may get inside the building, if it's too high the application performance may drop considerably. The first case can be used as an advantage when making reading on features like inside and outside.

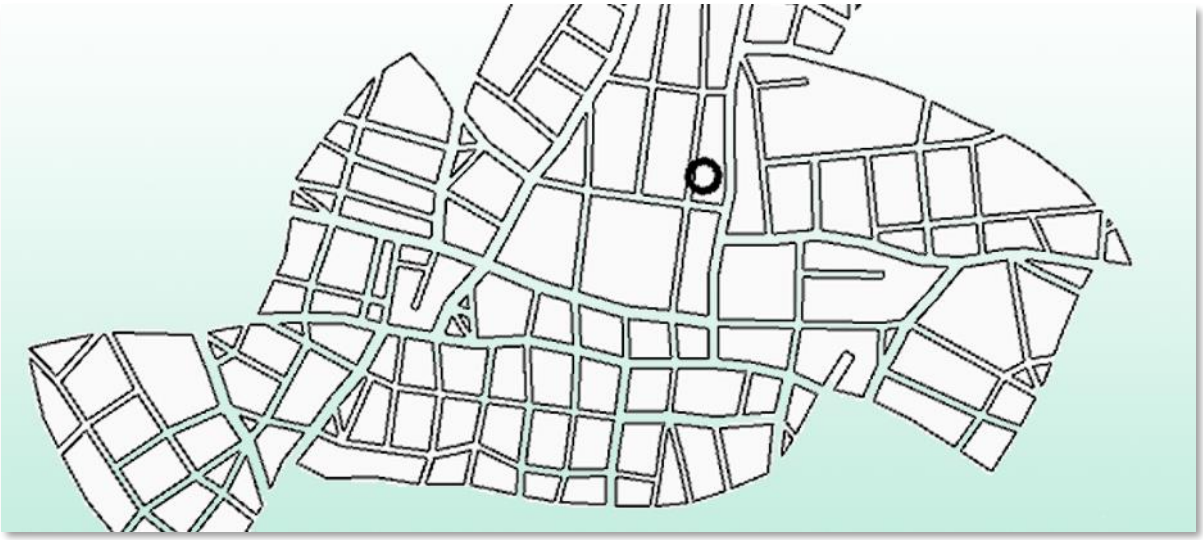


Figure 17 | Graphic representation of imported geometry

2 Object Avoidance

```
void Blocks() {

for (int i = 0; i < model.getSegmentCount (); i++) {
  Segment segment = model.getSegment(i);
  faces0 = segment.getFaces();
  for (int j = 0; j < faces0.length; j++) {
   Face f = faces0[j];
   vs0 = f.getVertices();
   // PVector[] ns = f.getNormals();
   for (int k = 1; k < vs0.length; k++) {
    // normal(ns[k].x, ns[k].y, ns[k].z);
    // vertex(vs0[k].x, vs0[k].y, vs0[k].z);

    Vec2D v1 = new Vec2D(vs0[k - 1].x, vs0[k - 1].z);
    Vec2D v2 = new Vec2D(vs0[k].x, vs0[k].z);
    float distance = v2.distanceTo(v1);

    float n = distance / 13; ////////////////

    for (int jj = 0; jj < n; jj++) {

     float x = lerp((float) vs0[k - 1].x, (float) vs0[k].x,
     (float) jj / n);
     float y = lerp((float) vs0[k - 1].z, (float) vs0[k].z,
     (float) jj / n);
     Vec3D v12 = new Vec3D(x, y, 0);
     vec3ds.add(v12);
```

Recognizes the imported geometry

Selects all the vertices

Organizes the vertex by number

In a 1 to 13 ratio (this value can be adjusted according to scale)

Repeats the process for y and z values

Figure 18 | Overview and code implementation of the object avoidance function

Attractors can either be positive or negative depending on the values being either positive or negative. This function defines an attraction force and a radius of influence around each attractor and can be applied to a set of agents which causes a reaction in a positive or negative way. These attractors can be positioned using the create attractor function. When positioned they act as static agents and comply to the same basic rules, scale of the force and radius of influence.

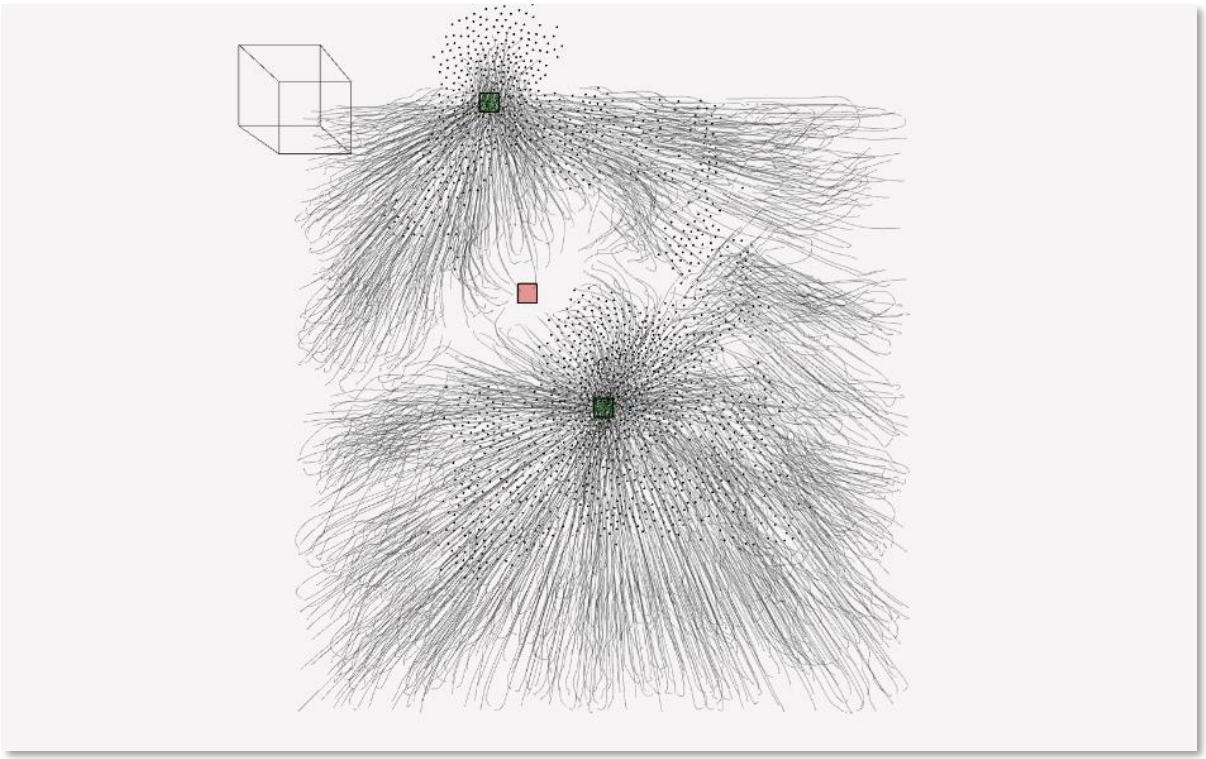


Figure 19 | Example image of a set of two positive attractors and two negative attractors

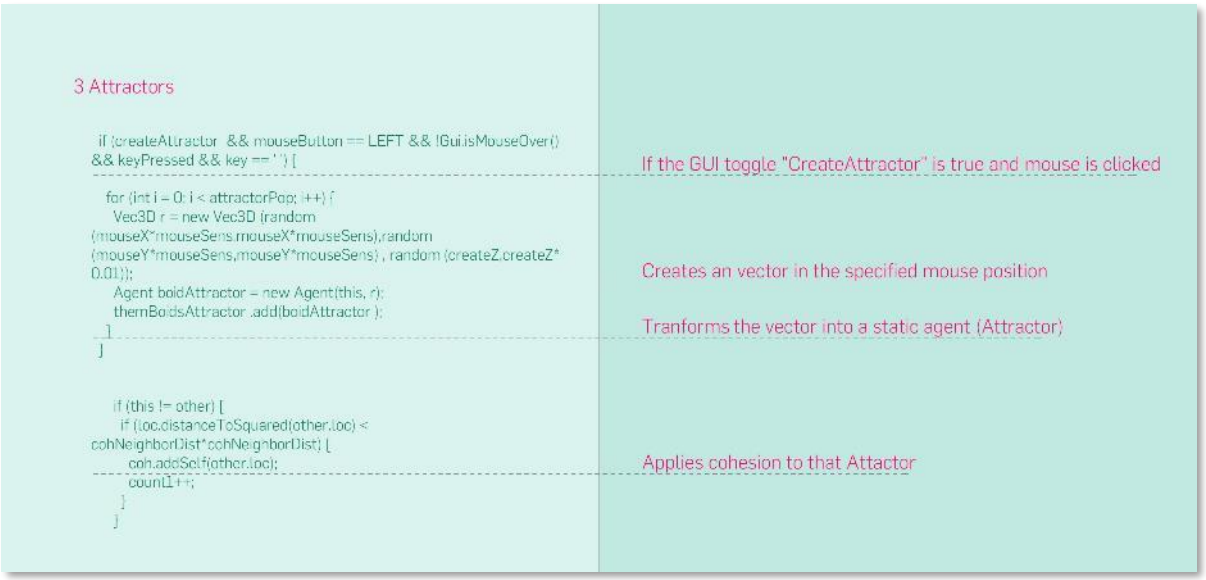


Figure 20 | Overview and code implementation of the attractors function

Path folowing is a function that was implemented in order to constrain sets of agents to a specific path. The way it works is each agent predicts its own future location based on its direction and speed at the given time. Then it calculates the shortest perpendicular line from the agent to the spline, which results in a point in that spline. That point acts as a force vector for each agent individually as a result agents start to move torwards that direction (each its own). When reaching this position, agents start get within range of each other and the flock function starts to operate. This is a stage where the decision in direction takes place, agents "decide" to follow a spline on one direction or the other depending on a number of factors namely: the angle that each individual reaches the spline, the amount of neighbours that are already following a specific direction and other external forces that are applied and in range (attactors, blocks, etc..). This is one of the core functions of this research, since it allows for performing simulations on context-specific grids and locations.

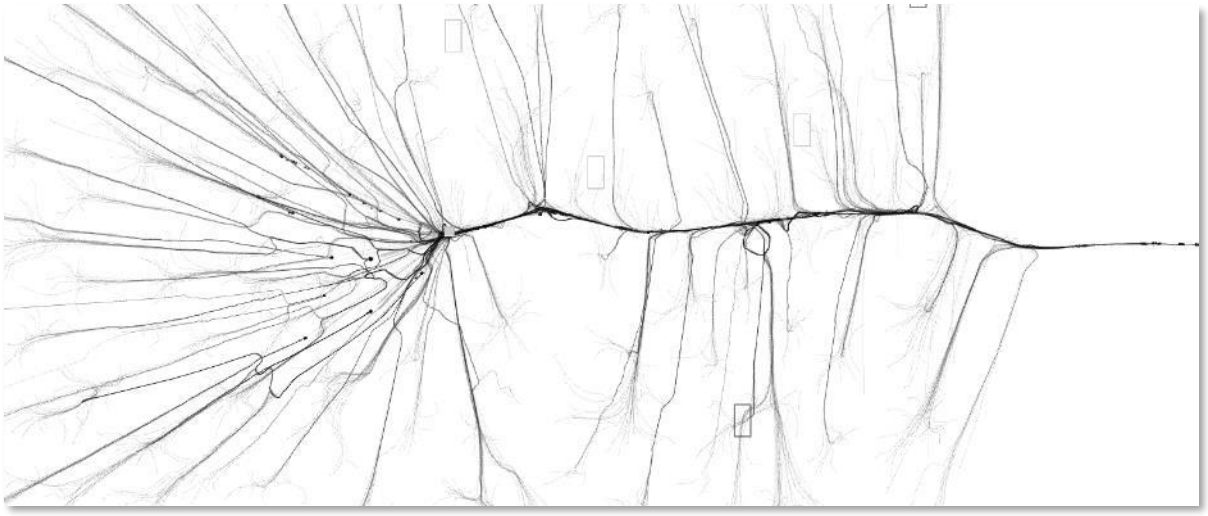


Figure 21 | Example image of a set of agents following a spline.

```
4 Path Following

Vec3D fLoc = boid1.futureLoc(lookAhead);          Predicts a vector with the agent's future location
  spP = new Spline3D();

  if (followSpline) {

  stroke(255, 0, 0, 150);
  boid1.vLine(fLoc, boid1.loc);

  Vec3D cns = boid1.closestNormalandDirectionToSpline(spP, fLoc, n);     Calculates the normal direction to the specified spline spP

  boid1.vLine(cns, fLoc);

  if (followSpline) {

  stroke(255, 0, 0, 150);

  boid1.seek(cns, followPath);
  }
  }
```

Figure 22 | Overview and code implementation of the path following function

Construct is a method that allows for the capture of certain moments in time with defined intervals by connecting agents and therefore writing it down in a specific plane. Each time that is applied (which is continuous), the z values of that plane increases with a specific value "incrementZ" through time. This function is combined with the nexus function (which connects neighbour agents accordingly to the specified proximity. Agent continue to move exactly the same way but in a different plane writing all the information. Also every time a plane is shifted a line can be produced to connect the same agent to the next plane. This is one of the most complex methods of creation, and possibly the one with the most potential since it allows for the creation of self-organized 3d shapes level by level with a well-defined structure.

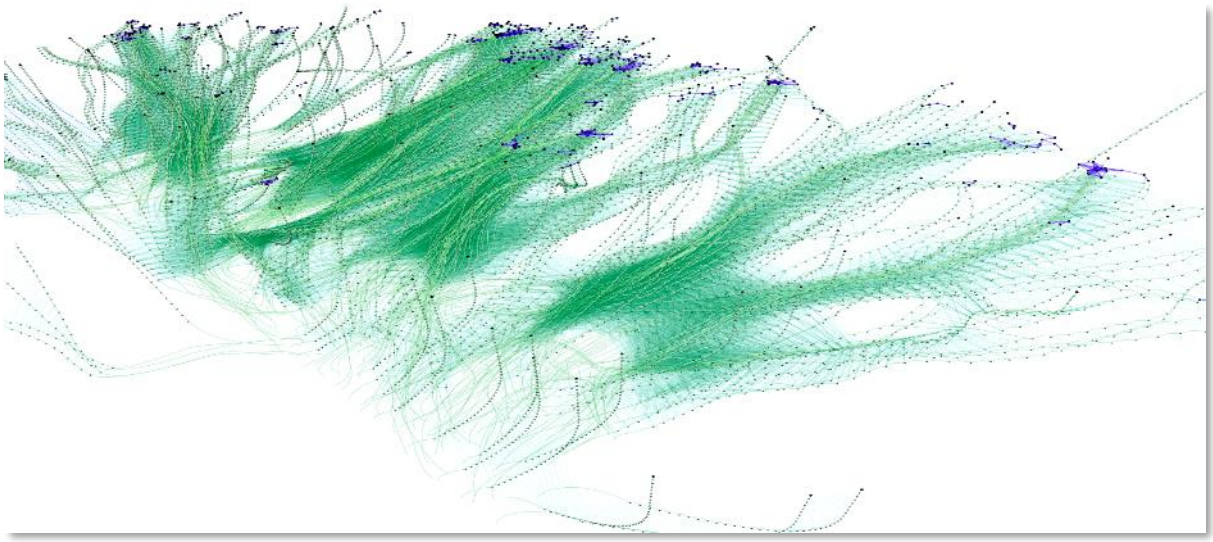


Figure 23 | Example image of a set of agents conforming to the construct method

5 Construct

```
void record (ArrayList<Agent> boids) {
  for (Agent a : boids) {
    pts.add(a.loc.copy());
  }
}

void connect() {
  for (int i=0; i<pts.size (); i++) {
    Vec3D p = pts.get(i);
    for (int j=i+1; j<pts.size (); j++) {
      Vec3D other = pts.get(j);
      if(p.distanceToSquared(other) < thresh*thresh){
      Line l = new Line(p, other);
        lines.add(l);
      }
    }
  }
}

void display() {
  for (Vec3D p : pts) {
    strokeWeight(8);
    stroke(155);
    point(p.x, p.y, p.z);

  }
}
if (construct && frameCount%5==0) {
  Frame f = new Frame();
  f.record(themBoids1);
  f.connect();
  frames.add(f);
  z+= (incZ);
}
for (Frame f : frames) {
  f.display();
  f.displayLines();
}
```

Records all positions of all agents and stores them

Applies a connection if within range

Draws the lines replicated

Draws the points replicated

Updates the Z position every 5 frames

Translates a copy of the object in the pre defined z value

Figure 24 | Overview and code implementation of the construct method

Stigmergy *"(...) is a coordination mechanism observable in biological insect societies like ant colonies. It is based on the behaviour of these ants to create a dissipative field by spreading smelling substances into their environment."* (Valckenaers, Kollingbaum, Van Brussel, Bochmann, & Zamfirescu, 2016, pp. 1-2), and was one of the last functions implemented into the developed application. The intention with the incursion of this naturally inspired processes into an algorithm and the relation with the subject of this dissertation was to improve consistency in the final representation of the extensive simulation results. Stigmergic behaviours allow for sets of agents to have, memory, in other words, to have a perception of paths that are generally more desirable consequently attenuating the tendency of these complex systems to generate chaotic and graphically imperceptible results.

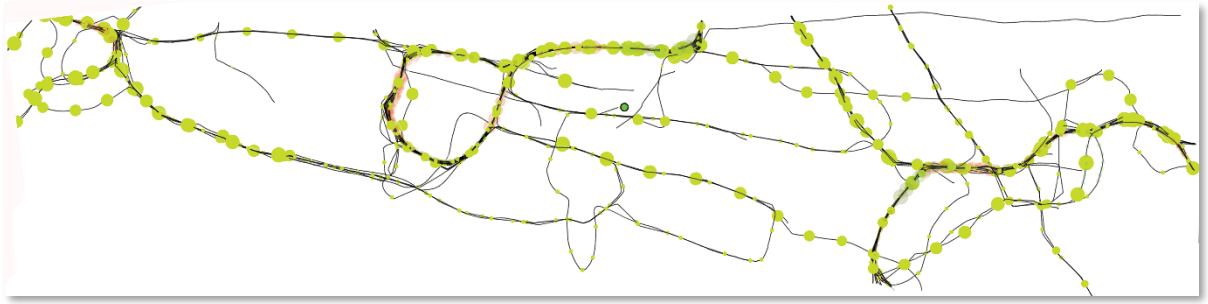


Figure 25 | Example image of a set of agents conforming to the stigmergy method

6 Stigmergy

```
Tcount++;
  if (Tcount > TStep) {
    tailF.add(loc.copy());                          Creates a tail every TStep frames.
    Tcount = 0;
  }
  if (tailF.size() > TLen) {                        With a certain number of tails per agent
    tailF.remove(0);                                Removes the first tail when limit is reached
  }

void tailCohesion(float magnitude, float range, ArrayList <Vec3D> field) {      Applies Cohesion to the tails

  Vec3D sum = new Vec3D();
  Vec3D steer = new Vec3D();
  int count = 0;
  for (int i = 0; i < field.size();i=i+10) {
    float distance = FutLoc.distanceTo(field.get(i));
    if (distance > 0 && distance < range) {
      tailPerip = (field.get(i)).sub(loc);
      tailAngle = tailPerip.angleBetween(vel, true);
      if (tailAngle < 0) tailAngle += TWO_PI;       Applies angle vison to each agent
      if (abs(tailAngle) < tailVAngle ) {
        sum.addSelf(field.get(i));
        count=count+1;
      }
    }
  }

 void tailSeparate(float magnitude, float range, ArrayList <Vec3D> field)     Applies Separation to the tails
{
  Vec3D steer = new Vec3D();
  int count = 0;
  for (int i = 0; i < field.size();i++) {
    float distance = FutLoc.distanceTo(field.get(i));
    if (distance > 0 && distance < range) {
      tailPerip = (field.get(i)).sub(loc);
      tailAngle = tailPerip.angleBetween(vel, true);
      if (tailAngle < 0) tailAngle += TWO_PI;       Applies angle vison to each agent
      if (abs(tailAngle) < tailVAngle ) {
        Vec3D diff = loc.sub(field.get(i));
        diff.normalizeTo(1.0/distance);
        steer.addSelf(diff);
        count++;
      }
    }
  }
```

Figure 26 | Overview and code implementation of the stigmergy method

3.1.2 Base Environment

This is the three dimensional environment where, in a final stage, the graphic results will be displayed on the screen. This embodies the virtual world where each simulation will be represented.

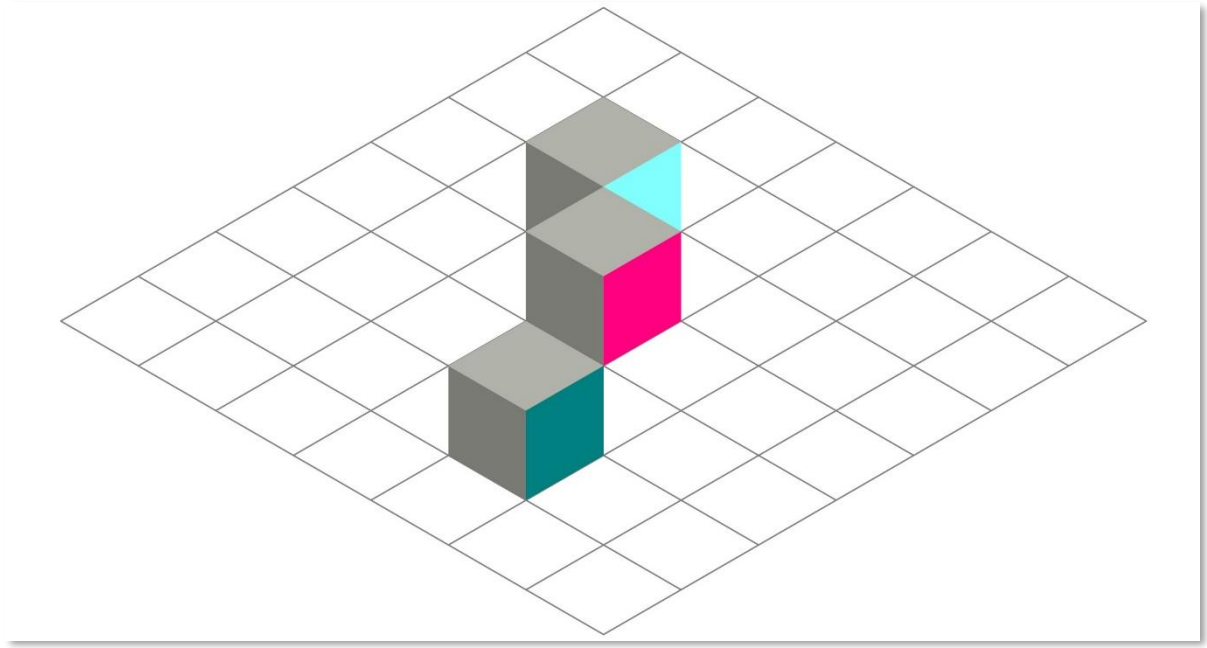


Figure 27 | Base Environment

This element acts as a canvas for every simulation. It's an infinite size three dimensional box in which every method function element and rule is either displayed, implemented or embed in. Everything created through "GUIcomponents" or imported via "ExternalGeometry" is actively displayed inside this screen. This is an environment with no scale initially is just an endless box. The notion of scale is only introduced when external geometry is imported and properly adjusted.

This function allows the user, when moving inside the "BaseEnvironment", to have a reference plane for creating elements in a "XYZ" coordinates. The "CellSize" – which is the size of each square of the grid - can be defined by the user. This function is directly related to the function "BOX" as is the two dimensional version of it being the only difference the "Snap" function

3.1.3 Graphic User Interface "GUI"

The Graphic User Interface (GUI) is the layer that is positioned between the user and program environment and facilitates the access to the features and parameters inside the actual program.

This section sums up the most important elements of the Graphic User Interface "GUI". This is one of the most important elements inside the application as they provide access to all of the Elements Functions and therefore Methods that can be used in the simulations.

The Graphic User Interface is composed by three main types. The “Slider”, the “Button” and the “Toggle” types.

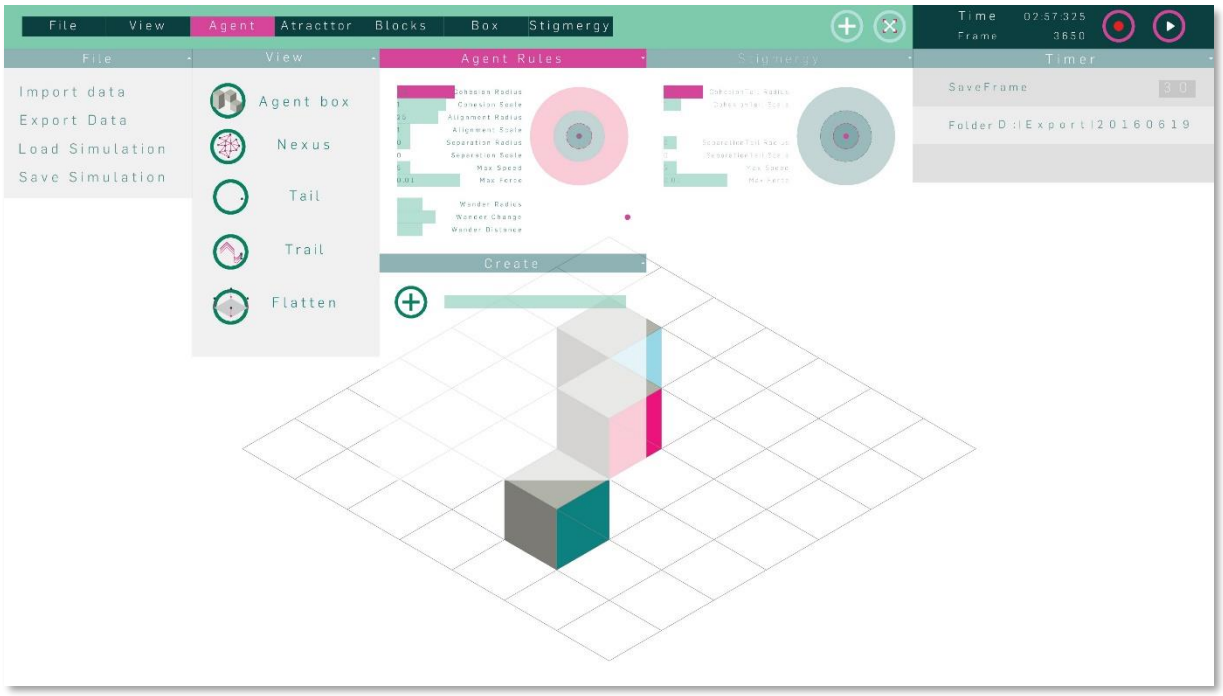


Figure 28 | General view of the Graphic User Interface.

Elements can be created and/or imported from an external CAD file so they can be used in the simulation. A graphic user interface will provide the ability for the user to have access to the functions inside the application in a fairly intuitive way. This component acts as a bridge to the computer’s root functions.

> “A computer program that enables a person to communicate with a computer through the use of symbols, visual metaphors, and pointing devices. Best known for its implementation in Apple Inc.’s Macintosh and Microsoft Corporation’s Windows operation systems, the GUI has replaced the arcane and difficult textual interfaces of earlier computing with a relatively intuitive system that has made computer operation not only easier to learn but more pleasant and natural. The GUI is now the standard computer interface, and its components have themselves become unmistakable cultural artefacts.” Steven Levy (2016) recovered from, britannica.com/technology/graphical-user-interface

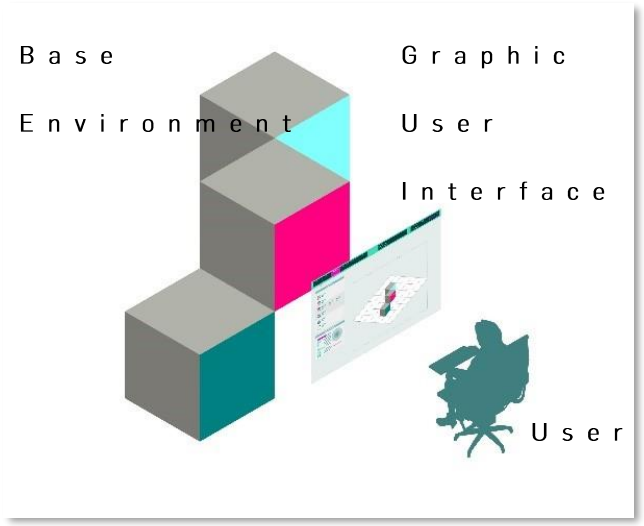


Figure 29 | Simulation components.

For the developed program the creation of a user interface was essential to provide a way to control the parameters and elements of each specific simulation and to provide access to the agent-based functions. A global overview of the GUI can be seen on Figure (X). The interface is subdivided into 3 main categories, the first has 6 main groups, File, View, Agent, Attractor Blocks and Box. The second group is dedicated to provide quick access to the most used functions and can be customizable by the user. Finally, the last group provides access to time-based features like saving screenshots or recording all frames in sequence, pausing the simulation and display time-based information.

In order to provide better accessibility, each function and variable is organized by category into six main groups called tabs.

Figure 30 | Main menu of the developed application.

In the example above the Agent tab is highlighted. That happens whenever a menu or specific function is selected. When that happens another set of menus opens. In his case, all lists related the Agent component open.

Figure 31 | Agent functions menu.

Each of these menus contain several ways to access the features, parameters and functions that are available for each simulation, in this case access to the agent rules.

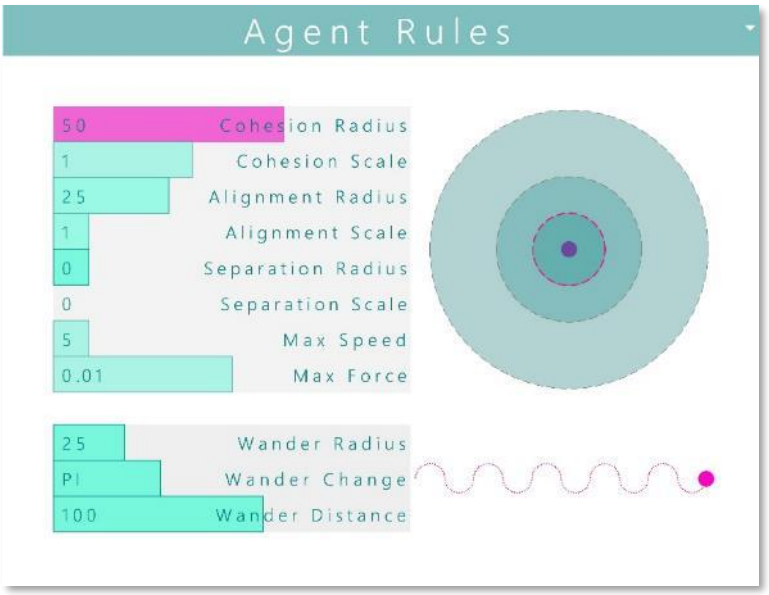
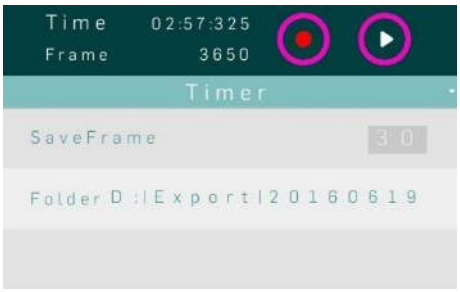
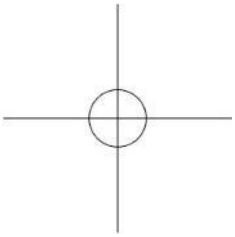
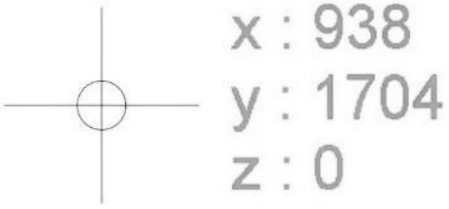
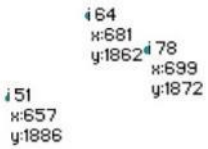


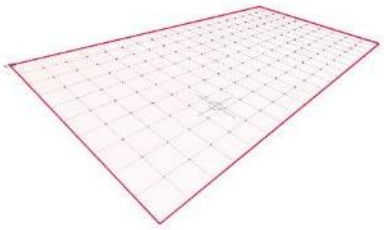
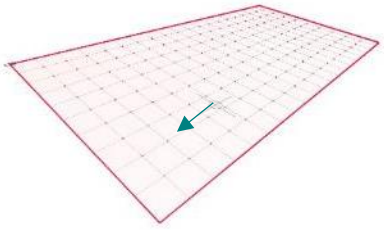
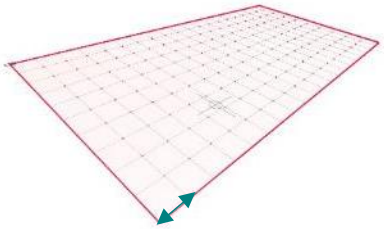
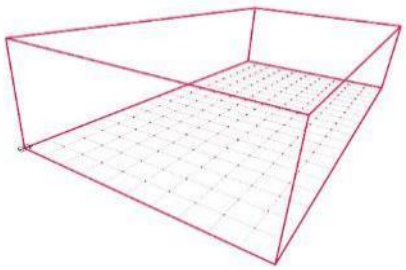
Figure 32 | Agent rules menu.

The slider component controls the values assigned in each function that can either be integers or float numbers. For example, to control a specific component like a box inside the base environment, one can use either a float or an integer, but in more demanding computational tasks like a dynamically changing value like a force or a distance between two agents it's highly recommended that an integer is used. This component plays a major role in the adjustments phase of the simulations, since a large number of complex operations that are dependent on these values, where a small change in a specific value, deviates the results in a very extreme way.

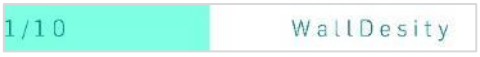
The next infographic table will provide an overview on the functions of the GUI and how they operate.

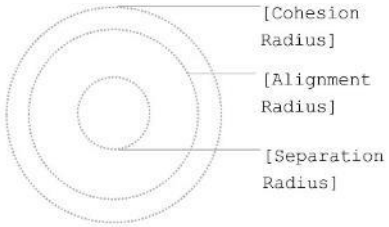
Function	Description	Interface element
General Interface and Base Environment		
Tabs	In order to provide better accessibility, each function and variable is organized by category into six main groups called tabs.	
Menu	Each tab can have a subset of menus that contain several types of GUI elements.	
Timer	In this menu all the function relative to time based method can be access and programed. In this example it show the save function and the specific folder that these frames will be saved.	
Slider	This type of GUI element allows for tuning variables and they can control floats and integers	
Toggle	These graphic user interface elements give access to certain functions on the application. They are on/off switches that activate or deactivate certain function that are intended to be operated. All the "Create" function utilize a "create" "Toggle" in order to operate. In this case the toggle "AgentAsBox" is represented.	
Button	Similar to "Toggle" elements these give access to a specific function only in a moment in time. When a task must be completed only once. One can think of it as a toggle that immediately turns off after being	

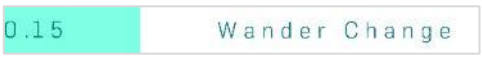
	used. An example would be the "ResetAgents" button.	
2D Cursor	This is the element that allows for the control and activation of each function inside the "GUI" and for the "Camera" movement. It's the arrow controlled by the mouse.	
3D Cursor	It works as the 2D Cursor but it operates inside the "BaseEnvironment" instead of inside the "GUI" and allows the creation of elements inside it. It's has a crosshair shape and it also allows to have access to some other useful functions like "ShowCoordinate" or "CreateZ".	
Show Coordinate	This function allows the user to obtain a specific coordinate "xyz" values of the "3DCursor" at a given moment accordingly to the current mouse position.	
Show Information	Show several types of information inside a specific agent. When active can show a circle around an agent with information like position, number of each agent, flocking values, forces that are being subjected to, visible neighbours and others. In this case, the number of each agent and its x and y position.	
Reset Values	Each value of agent rules are reset to the default values. These values are a good starting point to begin a simulation since they have a balanced "Cohesion" "Alignment" "Separation" ratio.	

Grid	This function allows the user, when moving inside the "BaseEnvironment", to have a reference plane for creating elements in a "XYZ" coordinates. The "CellSize" – which is the size of each square of the grid - can be defined by the user. This function is directly related to the function "BOX" as is the two dimensional version of it being the only difference the "Snap" function.	
Snap	This function calculates the closest "Node" from the "3DCursor" inside the "Grid" in order to use the "Create" function in an absolute coordinate.	
CellSize	Determines the size of each cell within the grid. This is also part of the "Creation" method described later on.	
Constrained Space	Inside the "BaseEnvironment" there is a "BOX" called "ConstrainedSpace" and it is the box that defines each simulation's boundaries. There are two function that are implemented into this function depending on the purpose "BounceSpace" and "WrapSpace" being the first the most commonly used. The image below is a clear example of a constrained space being used with the "BounceSpace" mode activated.	
Bounce Space	This functions applies to all dynamic agents and sets a rule to make each agent turn in the opposite direction every time it reaches the limits of the BOX of the defined "ConstrainedSpace".	n/a

Wrap Space	Not very frequently needed, but helpful in some cases, this method also applies to all dynamic agents and acts as a portal to the opposite face of the "BOX" in the "ConstrainedSpace".	n/a
Creation Functions		
Create Agent	Creates a set of agents in a specific position. The number of agents created is defined by the "AgentNumber" slider. It creates a certain number of agents at the desired location when it's "true", the space bar is pressed and the left mouse button is clicked. "x,y" positions are defined by the mouse, "z" position is defined by by the "CreateZ" slider.	
Create Attractor	This function allows for the positioning of attractors inside the "BaseEnvironemnt". This process is generally done in the "Setup" stages of the simulation but can also be done in real-time. This function can also be used and implemented as time dependent, each attractor can be active in a specific moment in time with a specific coordinate.	
Create Wall	Creates a shape initially composed by lines using the ""KeySpace"+"3DCursor"+"LMouseClick"" method. Creates one point each time this method is complete and connects the previous point to the next. After the shape is completed the key "Enter" can be pressed for the shape to be finalized. The other option is to press the key "C" so that the last point created connects to the first closing the shape. The last step in this method is to define the permeability of this shape created. The main purpose of this function is to act as a "wall" for the dynamic agents to not collide with. If any agent gets within a certain distance of this "wall" it does not go	

	through it. What was done was add permeability to this wall so the agent can go through depending on the permeability of the specified wall. This permeability can be defined by the "WallDensity" slider. These shapes created are the same types as the shapes imported from the external geometry after the import which can act as shapes "Block" which represent buildings and "Splines" which represent roads or paths.	
Wall Density	Defines the permeability of the shape being created. When creating a wall that acts as a block one can define its permeability to dynamic agents by adjusting this slider.	
Create Path	Splines can be methodologically defined as paths in which agents are confined to up to a certain level. In a real-life scenario they can be link to tendency tracks, like roads, sidewalks streets which have a higher level of usability. Also uses the ""KeySpace"+"3DCursor"+"LMouseClick"" method and allows for the user to manually create a path for the agents to follow. As the "CreateShape" draw function, this also uses the key "Enter" to finish the path and the key "C" to close it. It's important to understand that, although these components created or imported as splines define a tendency to use that path, depending on many other factors	
Create at origin	Continuously spawns agents at the origin "0,0,0". Can be activated using the "Key" "Q" or by using the "RandomAtOrigin" button. Nota that the origin coordinates can be defined previously.	
Create Radom	Randomly creates agents within a creation box defined by "xyz" values "RandomX" "RandomY" "RandomZ". This creation is continuous if activated	

	through the "R" "Key" and momentary if it's activated by the button "RandomAgents". Also has the option to create random agents inside the "Box". Notice that if "ConstrainedSpace" is active, the agents will be immediately confined to the "Box" that defines it.	
Flock Functions		
Flock Radius	This is part of the implementation of the cohesion rule explained in chapter 2 and defines a circle with a specific radius controlled by the controller with same name around an agent. The radius of influence of an agent can vary according to the specific simulation. Generally this value is the biggest of the three.	
Cohesion Radius	This is the implementation of the cohesion rule explained in chapter 1 and defines a circle with a specific radius controlled by the controller with same name around an agent. The radius of influence of an agent can vary according to the specific simulation. Generally this value is the biggest of the three.	
Cohesion Scale	This represents the strength of the force applied to each individual agent when within the circle of its neighbour. This rule makes the agents attract to one another if in range of its neighbour.	
Alignment Radius	The same method applies to the alignment. If a certain agent gets within the defined cohesion radius a force is activated in order for that specific agent align its movement accordingly to the nearest neighbour group. In general, this value is in between the cohesion radius and the separation radius.	
Alignment Scale	This represents the value of the force that is being inflicted on the agent within the in range radius.	

	This makes the agent act in a more natural way before the separation is activated.	
Separation Radius	The same principle is applied to the separation. Each time an agent gets too close to its neighbours, a repulsive force is activated. This is generally the smallest radius of the three.	
Separation Scale	Generally this is a relatively higher values than the other two rules although it really depends on the intention and strategy of each simulation. If the intention is to write the trail information of a group as shown later on chapter 3 (section 3.3 Integrated Simulation) this value can reach values close to zero. This will demonstrate that a large group of agents have wrriten information on the same place, this is translated in a more pronounced line weight therefore it has a more relevant result.	
Max Speed	This slider defines the maximum speed that every agent can achieve. This variable has a reflective relation with alignment since the latter is intended to adjust and normalize the general group speed. The speed of every agent can be actively visualized in the following graph in order to get a better perception of the global groups that can be sometimes emerge. Generally higher graph values mean higher alignment local values as this graph represents the current speed of each agent and as speed is directly dependant on alignment radius and scale.	
Wander Radius	Max radius of the circle applied to the wander function. Bigger radius means a bigger circle for the path of the agents that is being applied to.	
Wander Change	The ratio that defines the randomness of a variation in direction.	
Wander Distance	Minimum distance that an agent travels without changing direction.	

Graphic Functions		
In order to visualize the global behaviour of the set in each simulation, different types of representations are implemented, each one with its specific qualities. Through this representations one can have a better reading of the simulation all the way through. These functions can have different methods of visualization and represent different qualities like proximity relations tracing paths, and others.		
Nexus	This visualization function represents graphically with a blue line agents that are within a defined range of each other. This can translate proximity behaviours. It is useful when adjusting the basic low level rules specially the ratio between Separation and Cohesion. On the first images different ratios between Cohesion and Separation can be adjusted in order to define the right neighbourhood for each set of agents. This function can be also applied to blocks. In this case a proximity line is created between the dynamic agents and the block static agents according to a nexus value. This allows to have a visual perception on proximity relations between the agents and the blocks.	  
Bezier	This display function creates a curve defined between two or more agents that are within the nexus range. The size of the curve is defined by the agent velocity.	
Bezier Between Tails	These are all dependent on 2 types of values named nexus values and they have a minimum and a maximum value for the distance: MinDistance and Maxdistance “Variables”	
Trail	Draws a path of each agent during the simulation in a line format.	

Tail	Drops a point periodically in the space for a limited amount of time (frames). Is dependent on these two values: TailNumberOfPoints TailUpdate	
Save Frame (screenshot)	Saves a raster image to the application folder "Screenshots". This is the quickest way to get the image on the screen. The resolution on this image is always limited to the resolution of the screen that the application is running. Can be activated through the "P" "Key"	
Make Movie	Works the same way as the "SaveFrame" function but acts continuously. Saves a collection of sequenced images from the moment that is active to the application folder "Frames". This sequence will allow for the creation of video sequences after the simulation ends. This function is heavy on computer resources and has to be used carefully. It requires a lot of disk space and RAM.	
Save PDF	Saves a vector image to the application folder in a PDF format. Because the "SaveFrame" function is limited in terms of resolution, this function was implemented in order to get a vector image. There are some elements of the screen like the GUI that will not be shown on this type of export. Because this format does not have a defined scale, the thickness of each line might not be what can be seen on screen, but is always accurate in a relative way. Can be activated through the "S" "Key".	
Export DWG	Saves a 3D cad file to the application folder in the DWG format. (Experimental, unstable). Some elements like points or splines are not directly compatible with the DWG format. Any points represented in the simulation will not be exported or will cause the application to crash.	
For easier access to each functions some keys are implemented as shortcuts either from the keyboard or from the mouse: "SHORTCUTKEYS" "\" Activates menu 0 "Base"		

“1” Activates menu 1 “Agent” “2” Activates menu 2 “Blocks” “3” Activates menu 3 “Attractor” “4” Activates menu 4 “Terrain” “5” Activates menu 5 “Box” “Q” Activates “CreateOrigin” function. “ERASEAGENTS” CREATION FUNCTION Erases the set of agents created previously. Can be useful when doing several simulations after setting up the static elements in the environment especially at an early stage when the values are being adjusted. “KeyBackspace} “Navigation” In this section all the navigation functionalities will be covered. These allow through the Camera function to move inside the “BaseEnvironment”, most of them are self-explanatory. “Camera” The camera function serves the purpose of moving the viewpoint around within the “BaseEnvironment”. “Pan” Pressing the middle mouse button and dragging the mouse allows for panning the camera. “Rotate” Left mouse click and drag allows for rotation around. “Zoom” Using the mouse wheel controls the zoom level of the camera “CamPosition”

Figure 33 | Summary table of the most relevant functions of the developed application.

This concludes the description of the functions controlled by the graphic interface. These functions were the most relevant for the use of this application in this dissertation. They allow to create adjust and activate certain processes through the user interface into the base environment. The previous table acts as a guide for the user to have a notion on what function specifically being controlled or activated and at the meantime summarizes the structure of the application.

4 Model Application

In this chapter, the developed methodology assembled in the prior, will take a practical approach to most of the concepts that helped this assembly. This serves the purpose to demonstrate the potential response to the objectives of this dissertation as well as its potential in a future development.

The main focus is to consolidate the multi-agent-based-system application and understand how it can operate, from the most basic low level operability that contains the native components which are the individual elements that compose the structure of the program, to the next complexity iteration that is the interoperability between these native components, external components and user interaction and lastly the systemic interoperability that sums up all elements functions and methods.

In addition, the structure of the application itself is going to be clarified, from the basic building blocks which are individual elements that compose the structure, agents being the most important, to attractors, paths and tasks which allow for the user to access certain functions of the software ending in a systemic which are then condensed to methods that let achieve certain results that combine these 3 main application components. It will also be explained how the graphic user interface will allow access to each of these Methods Functions and Elements to construct a methodology of operation.

This chapter is divided into three stages where in each stage different approaches to the simulation are implemented. These stages played a major role in the development of the application since through experimentation they provided focus to what was intended to be achieved. The first was highly experimental and tried to demonstrate the potential of the application. The next step was to experience the application in an architectural way and decide which functions could be implemented in order to fulfil the needs of urban design. Lastly, the final step, where the application is closed and it's tested in a real-world scenario and is combined with other methods.

4.1 Defining a strategy

At a starting point one must adopt an initial "Strategy" accordingly to the intended simulation "Objective" and filter every aspect that is considered imperative to each simulation.

An initial strategy will take place before each simulation in order to:

- Focus the simulation on a particular concept that is to be studied.
- Simplify the translation of context specific elements. (attractor, paths etc.)
- Be systemic and don't change too many variables at once.

Although these principles are useful, it is important to notice that a simulation can be time-consuming. The time it takes always depends on what is the focus of the simulation and the understanding on each variable. The adjustments made when preparing a simulation must be sequentially incremented in order to understand how that specific variable is affecting the system in a global way.

After extracting the results of each simulation one must approach those results and determine if the readings are substantial accordingly to the preliminary objectives. If the results are not conclusive according to the "Objective", "Adjustments" must be made in order to fine tune the next iteration of the "Simulation". This continuous learning process allows for a better understanding on how each specific system works for a particular space and how each variable and rule interacts with the surrounding environment. If after several iterations none of the results are not assumed "Conclusive" one must redefine the "Strategy" taken beforehand and run the whole process again.

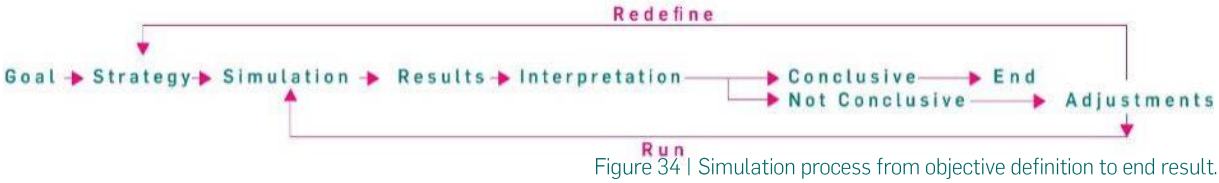


Figure 34 | Simulation process from objective definition to end result.

The results following this strategy are selected and represented in this chapter and were selected using the following criteria:

Consistency

The selected simulations must have consistency in its results, meaning, the results extracted after each simulation with the same exact variables display similar results in general. This provides a way to dissolve the random factors that the swarming algorithms have in its nature while still making use of them consistently.

Relevance

Each simulation is selected based on relevance of the theme being studied. Meaning that a simulation can only be selected if it produced results that have specific emergent qualities that justify this selection. Each “relevant“ simulation is only relevant if it’s unique in its nature this lead to the next criteria Variety.

Variety

Every selected simulation has a specific keyword or theme to differentiate them from each other in specific themes in order to ensure this variety. A set of keywords will be attributed to each simulation in order to catalogue them and single each other out.

Logical graphic representation

The results produced must have a clear graphical representation of what specific feature is being addressed. The colours of the attribute that is being studied must be differentiated as well as the scale that it’s is being represented.

These guidelines provide orientation in the process of selecting the results that are significant for this dissertation and simplify the process of comparing simulations with other ones similar in nature. Note that these selections summarize the most relevant results within a much broader list of graphic representations.

4.2 Unstructured Simulations

The following sequence of images represents the process of development of the application in a chronological way. And the most important features of each stage will be described, the methods and functions used as well as elements and the initial strategy.

Although the experiments in this section were highly unpredictable – since they result of intricate emergent events that cannot be predicted in an initial state – they allowed for further understanding on how its elements operate thus pushing forward the development of the application itself.

The main goal is to allow the user, after running through a set of trials, obtain shapes and concepts that will enable for a further understating of certain qualities of specific spaces in an abstract way for further interpretation and translation. After this, one can translate these shape behaviours to “real-world” concepts like flow, tension, looseness, proximity relations and others.

Urban like patterns

This set of images represents an early stage example where the introduction of the attractor concept was implemented along with the other flock rules. Randomly distributed agents inside a "Box" try to find their way into the most attractive place on a "ConstrainedSpace" and negotiate their individual position among themselves.

Methods and functions:
Initial location – Evenly distributed in order to get consistency throughout the simulation iterations.
Cohesion – High cohesion values allowed for highly defined tendency paths.
Alignment – Medium alignment values allow for smoother paths and space negotiation.
Low Separation – Low separation values allowed for highly defined tendency paths in open spaces but also a more aggressive local negotiation in constrained highly desired spaces.
Attractor – Two attractors with moderate values
Negative attractor – One negative attractor with a high value.

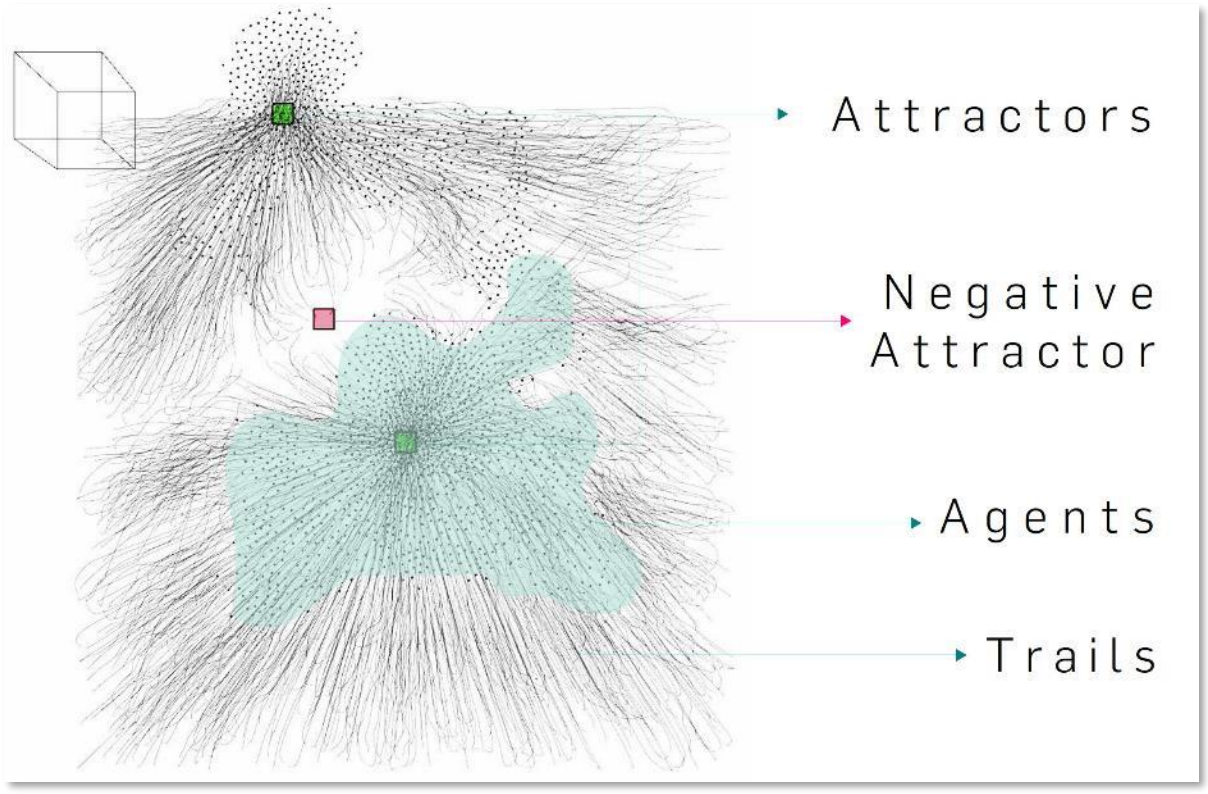


Figure 35 | Example image of a set of two positive attractors and two negative attractors.

At this stage the patterns that were produced started to represent some similarity to emergent urban patterns. From this point on the development of the application focused on achieving and seeking this similarity in order to be able to explore the potential of recreating behaviour that were similar to real world scenarios.

As the agents progress through the simulation they leave behind a “Trail” that allows us to perceive patterns that graphically bear a resemblance to urban-like patterns of human locomotion.

At the beginning of these simulations, a set of 2 negative and 2 positive attractors were created randomly across the box environment, as well as a set of 1000 randomly distributed agents while making use of the function “Trail”. The condition for this functions is that length of each trail is limited to a number of frames. This provides consistency across other simulations that poses the same set of variables. Therefore, after the same amount of time each simulation ends. This frames that were extracted from the simulations represent the moment were the simulation ends.

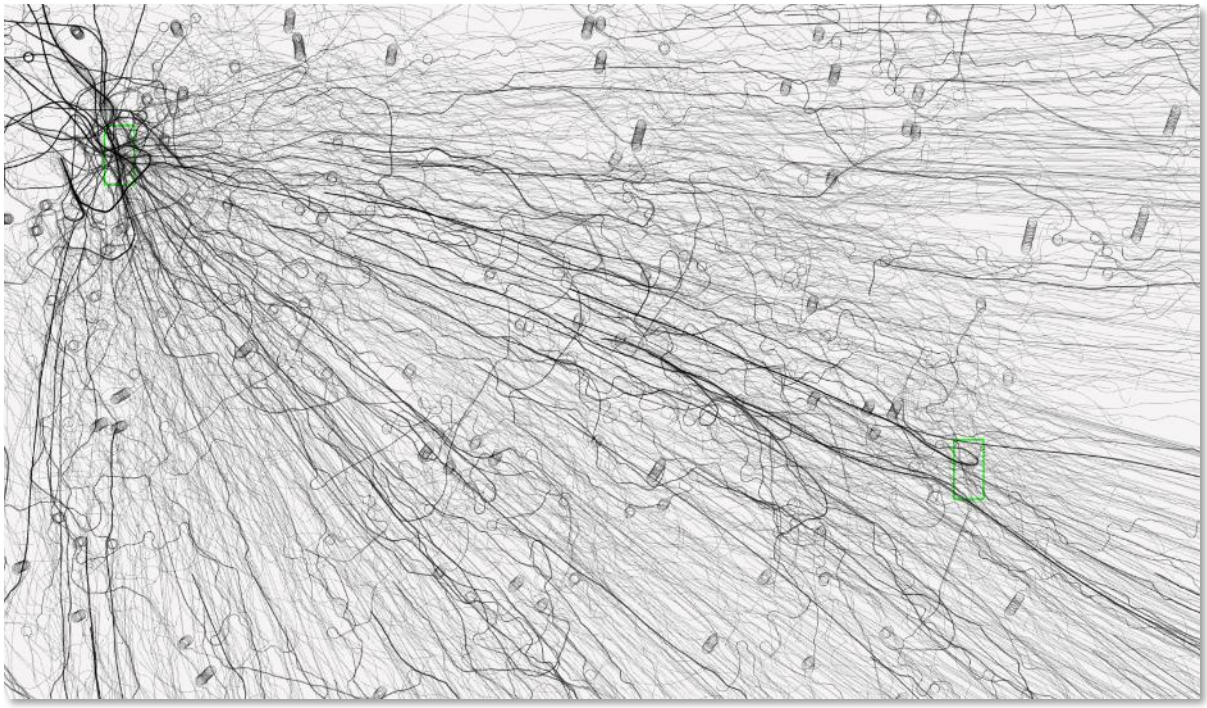


Figure 36 | Example image of a set of agents with two attractors.

At this stage the emergent patterns produced started to represented certain some similarity to emergent urban patterns. From this point on the development of the application focused on achieving and seeking this similarity in order to be able to explore the potential of recreating behaviour that were similar to real world scenarios. As the agents progress through the simulation they leave behind a “Trail” that allows us to perceive patterns that arguably start to resemble urban-like patterns of human locomotion.

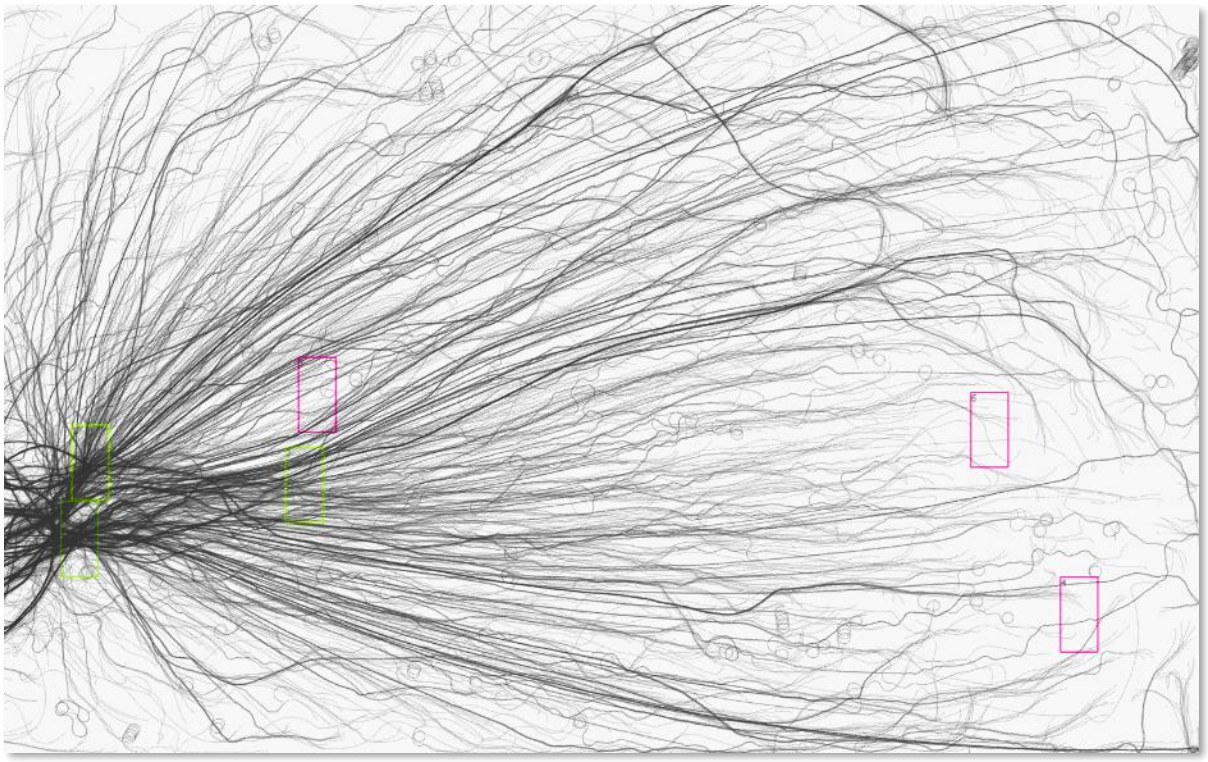


Figure 37 | Example image of a set of two positive attractors and three negative attractors.

After a set of trials, it was concluded that these pattern were heavily dynamic in form as opposed to some urban grid examples and the similarities were based exclusively on space negotiation.

Figure 38 | Example image of an existing urban grid.

This example urban grid is overlaid on top of the simulation and provided a notion of what was intended to be achieved. The selection of this grid representation was made to provide a middle ground between the goal of these simulations and the results that were actually being produced.

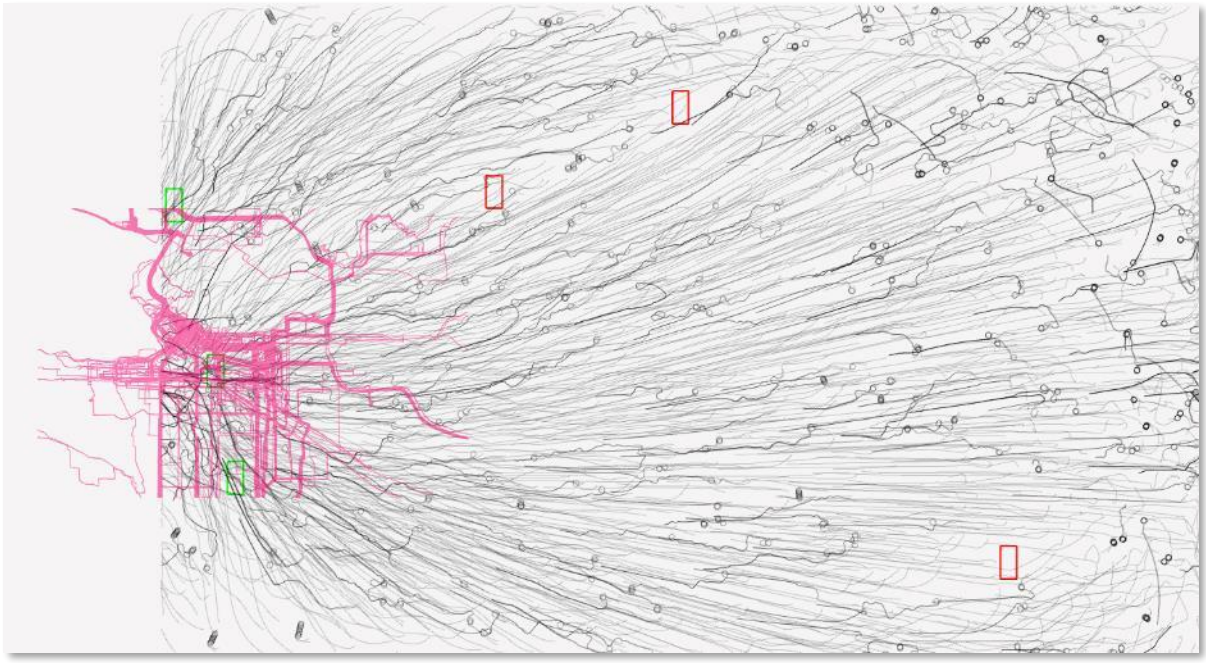


Figure 39 | Example overlay image of an existing grid with a simulation with three positive and negative attractors.

This led to the implementation of a function that would allow to “create” and “read” external CAD geometry inside the application. The next example shows the first stages of that implementation that would allow for certain existing paths to be taken as a priority. In this case, a randomly generated line was created each time in the middle of the environment where a “Seek” function to that spline was implemented. This “Seek” function acted in a similar way as the positive attractor function with some dissimilarities. Rather than seeking a specific positive attractor, a calculation was made to the closest normal point between the spline and every single agent.

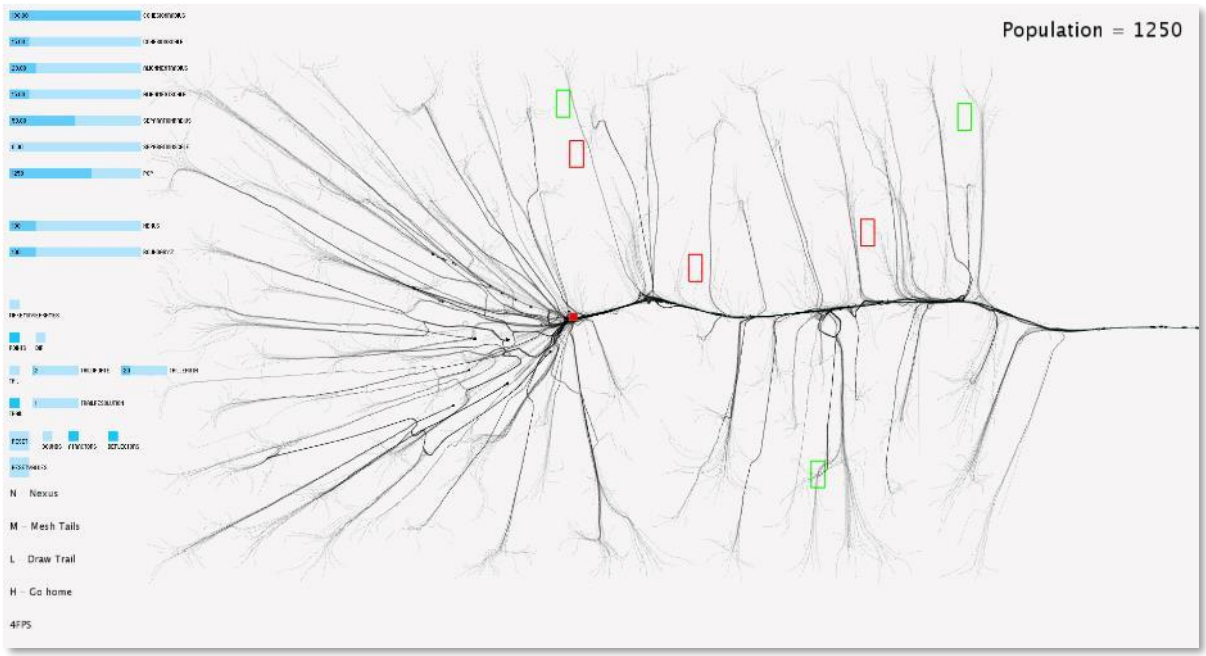


Figure 40 | Example image of a set of agents reacting to the follow spline function.

At this point, agents, when subdued to this new function, started to ignore positive and negative attractors almost completely and started to follow the spline exclusively. This led to the conclusion that some variables had to be adjusted in order for them to work together rather than against each other. A set GUI element were then introduced in order to fulfil these adjustments.

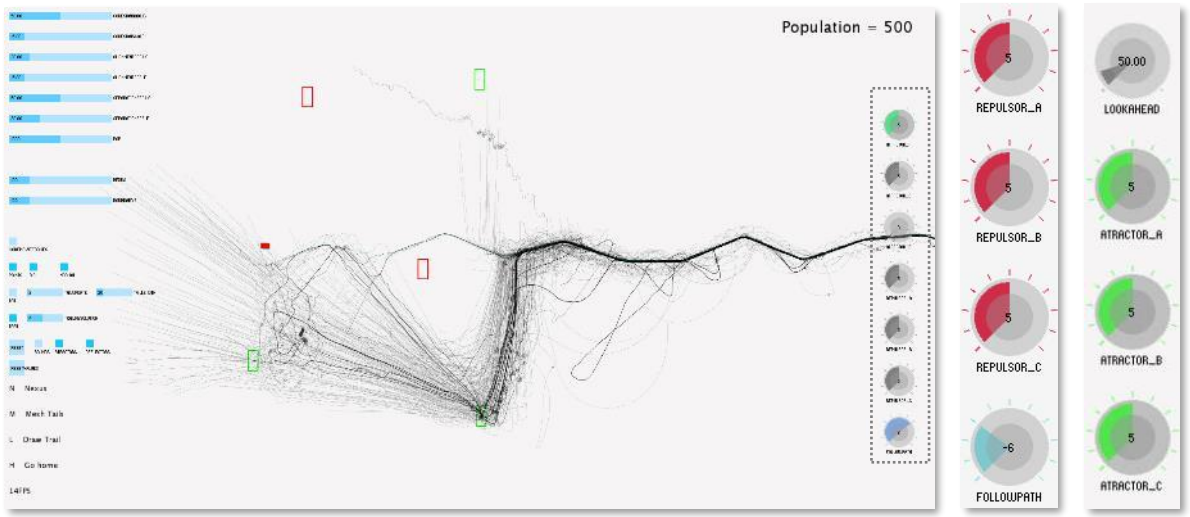


Figure 41 | Example image of a set of agents with balanced forces between the spline and the attractors.

After making these adjustments agents were able to be follow the defined path while at the same time being affected by the positive and negative attractors. This step provided a way forward in pursuing one of the objectives of this dissertation which is being able to simulate according to context specific scenarios and also led to the next stage of development.

4.3 Drawing implementations

At this point of the application development, a basic drawing functionality was introduced in order to be able to draw on top of imported cartography which also led to next set of simulations in square in Porto (Praça da Batalha) which acted as a testing bed for the drawing functions to be verified.

The way this drawing function "DrawSpline" operates is similar to any kind of CAD software, which starts by picking a set of points in order to create a shape which can either be a path or barrier where paths represent specific lines that agents can be confined to or, barriers with different permeability values, that represent buildings or sidewalks.

The differentiation between barriers with more or less density allowed for the creation of a hierarchy where places that were dense like buildings (red) could not be crossed and places like roads (green) could be crossed but with a certain drag on agents.

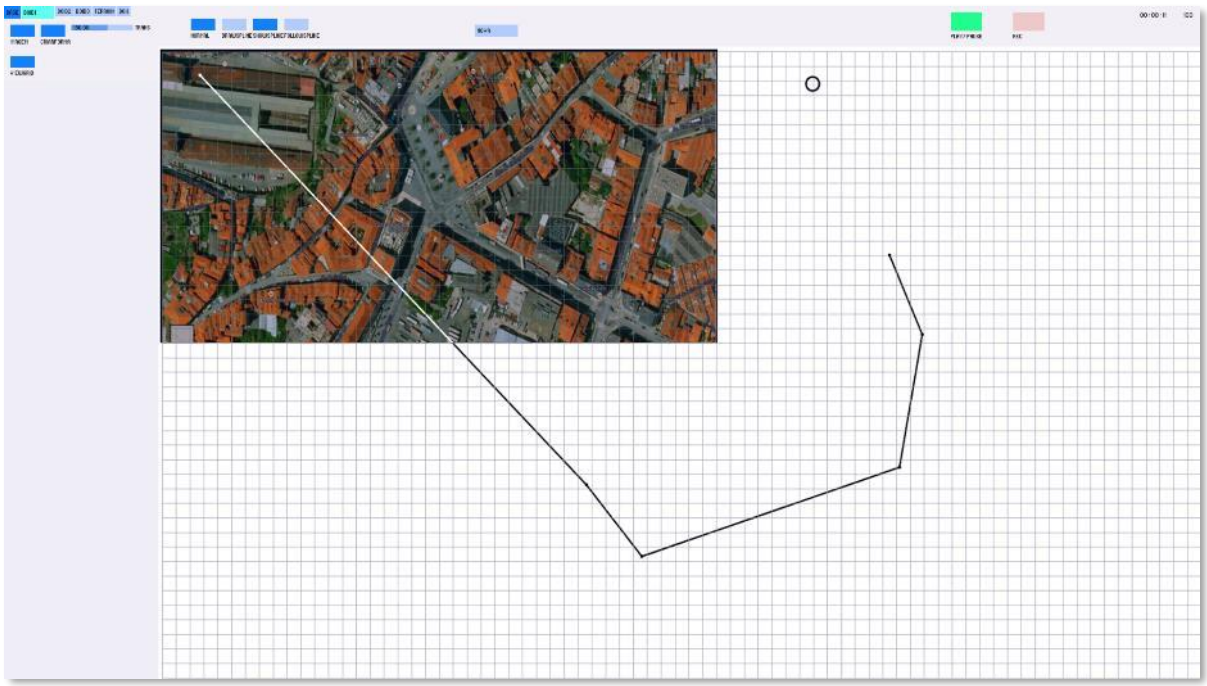


Figure 42 | Early stages of the implementation of the draw function.

This run focuses mainly on acquiring an in depth understanding on how components like barriers with various permeability values work with different levels of attraction to a specific block. It also lets some functions of proximity to become active as well as the path tracing function. This test has no defined physical scale and it was meant to understand how some components react with each other.

In this simulations the variables and conditions created were defined to represent pedestrians moving in accessible space. Some areas, were more accessible than others and barriers with different levels of permeability were implemented like sidewalks and roads.

Figure 43 | Draw function implemented in an imported map in Praça da Batalha.

The image below represents an applied cumulative tracing of the path created by agents while reacting to the defined values of density for roads and buildings to simulate flux in a square in Porto (Praça da Batalha) and their ramifications. This is a flux representation of a series of agents roaming around the accessible areas of the square and the streets around it. Some areas were more accessible than others depending on density. These results are relative in terms of scale and the rules that are being implemented and the adjustments of the variables.

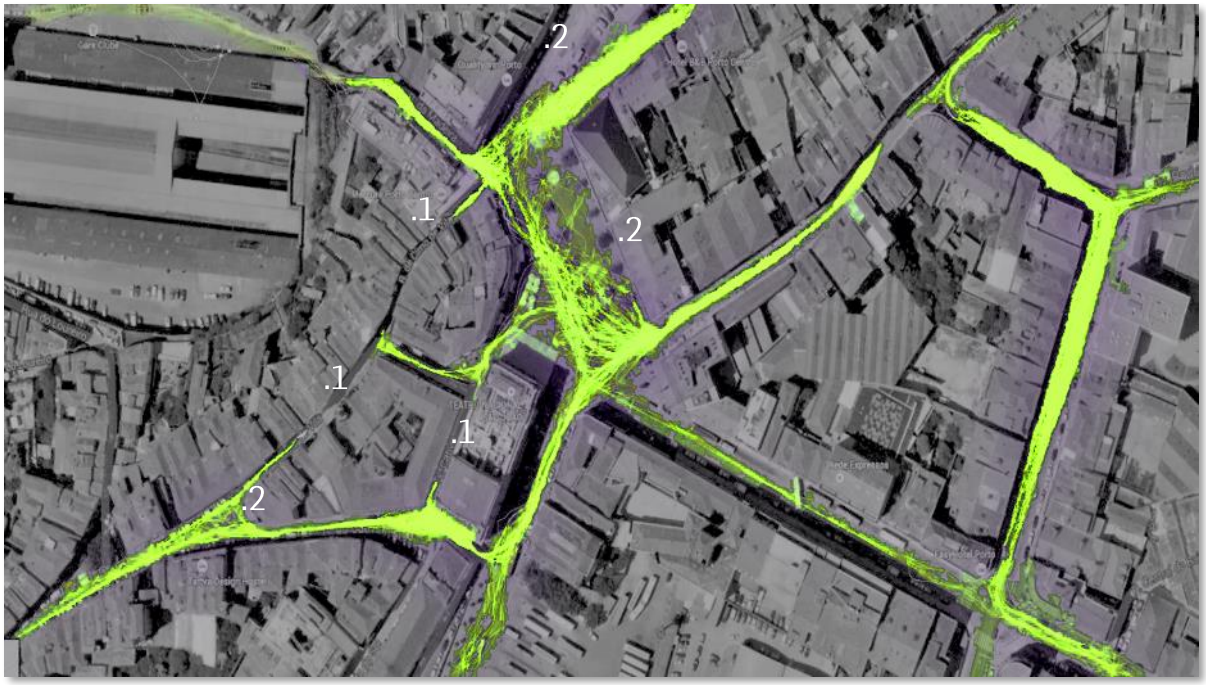


Figure 44 | Early run of a simulation in a context-specific environment (Praça da Batalha).

This stage allowed for the application to be able to simulated how constrained spaces and open spaces differ in terms of general agent reaction, the tendency of groups to pushed out of narrow streets and areas that are not wide enough for natural motion (.1). On the contrary, open spaces like the square itself allow for a more fluid negotiation of space thus allow for a more “friendly” motion (.2).

In this simulations the variables and conditions created were defined to represent pedestrians moving in accessible space. Some areas, were more accessible than others and barriers with different levels of permeability were implemented like sidewalks and roads.

The next series of images represent the stage where imported external geometry function was implemented. The purpose of this run was to apply the agent-based method to a non-context-specific model. These are structured simulation results. Structured as they follow a specific external geometry either created inside the interface or imported from a third party program.

This was the moment of the development of the application where external imported geometry in a CAD format was implemented. After the implementation of “importDWG” function of a hypothetical urban grid (externally produced and procedurally generated) was used as canvas.

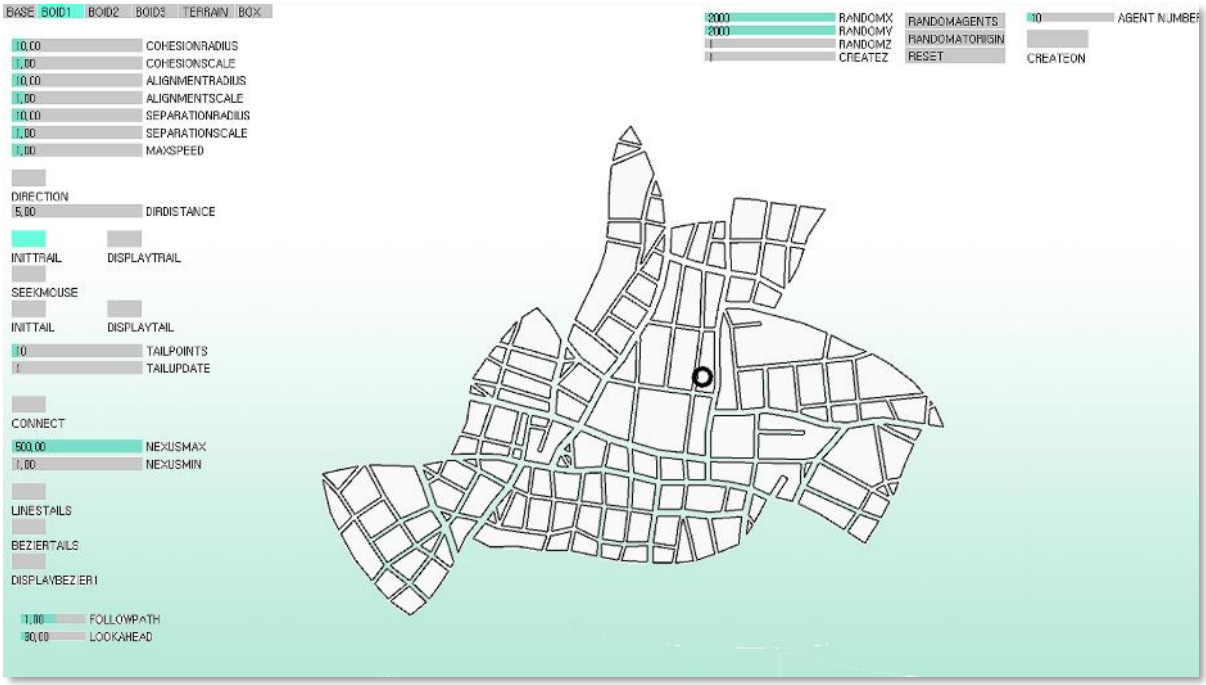


Figure 45 | Example image of the imported external geometry function at early stages.

The first step was to import the plan in a Wavefront .OBJ file and adjust the scale to fit the program workspace. It is important to mention that a Wavefront .OBJ file is in an open-source format based on a .CSV (Coma Separated Values) file format meaning that it can handle 3d geometry like vertices polygons faces and other types. This kind of information is imported in an xyz format into the program then translated into the geometry of the roads and buildings.

The image below represents the set of points imported from the externally generated plan that were later used to construct geometry inside the base environment.

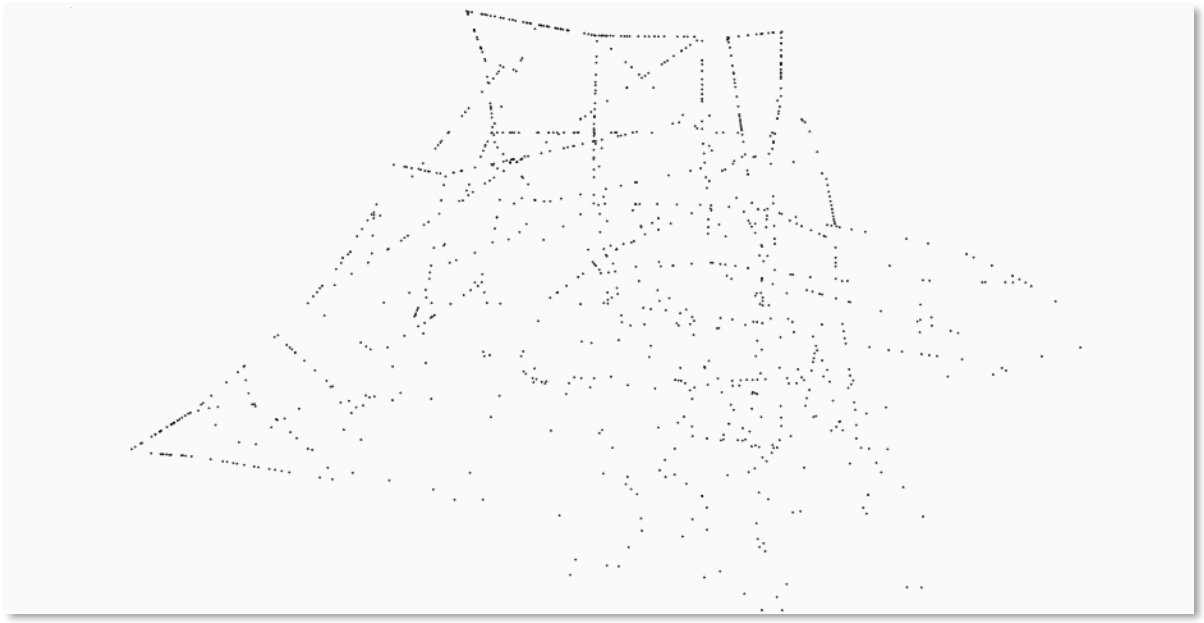


Figure 46 | Points imported from the externally generated plan.

Set of points imported from the .OBJ file and translated into application points.

The purpose of the next series of test simulations was to adjust the system itself to be able perceive reaction flows and behaviour of agents between themselves and their environment following the specific lines of imported geometry.

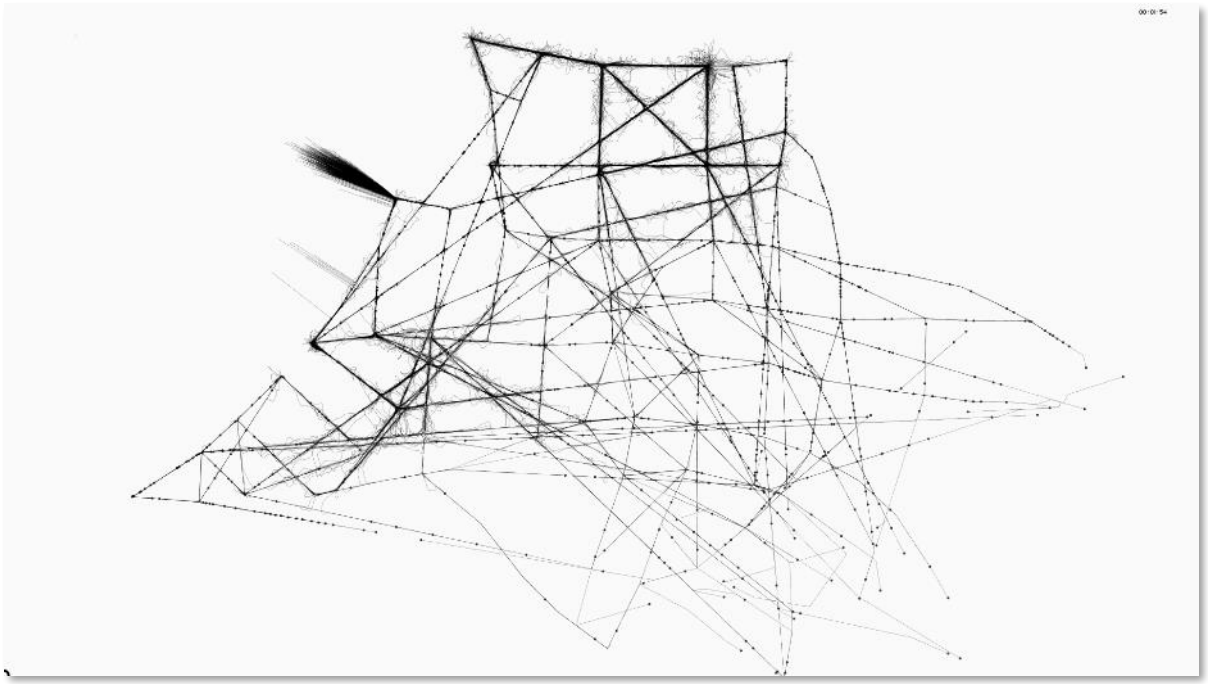


Figure 47 | Example image of an early stage simulation with the imported geometry function implemented.

The strategy in this simulation was link two concepts, flux and mass, which relate directly to proximity factors. This in a later instance would translate to street angles and curves. Some tuning of the visualization method was made in order to explore these concepts like the use of trail and tail in order to represent these concepts. As seen on the next image, agents are confined to the specific imported grid while still making some exploration in its surroundings.

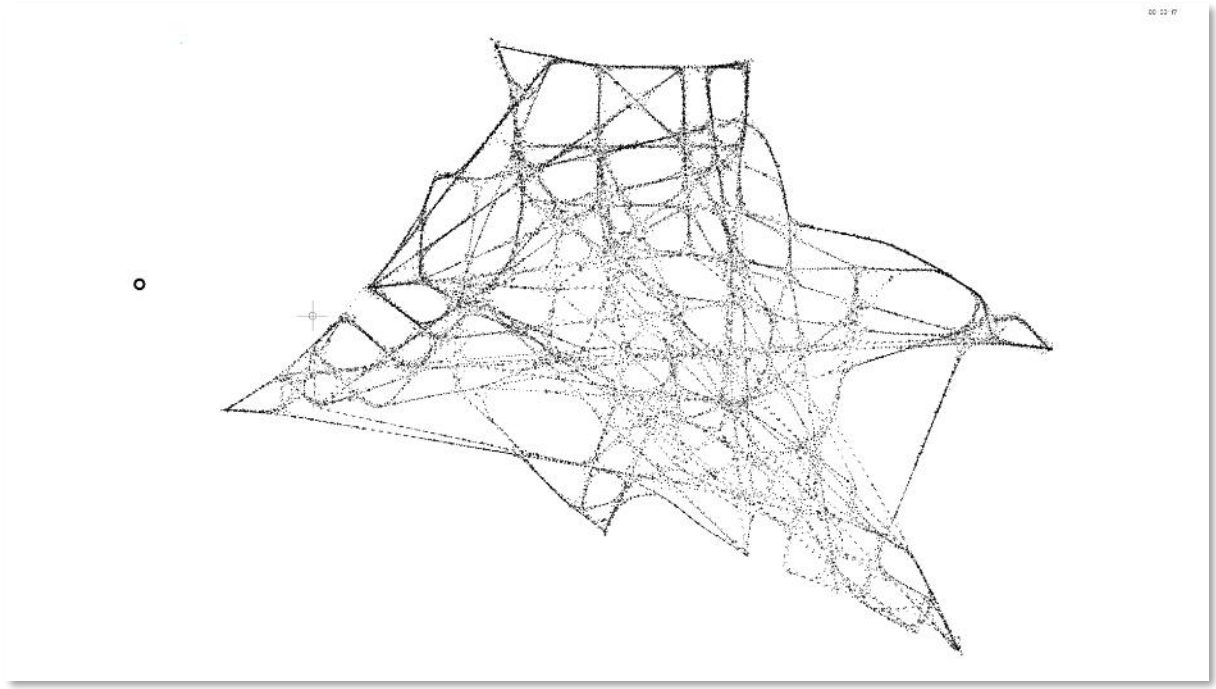


Figure 48 | Example image of an early stage simulation with the imported geometry function implemented with tail funtion.

Meanwhile some of the graphical function were explored in order to get different types of readings. The next image shows some density values were agent tend to concentrate in specific nodes that connect the imported paths, this function will be relevant when studying density in real world scenarios. After fine-tuning the behaviours some visualization methods were activated, explored and fine-tuned in order to have a better understanding on what is going to be simulated.

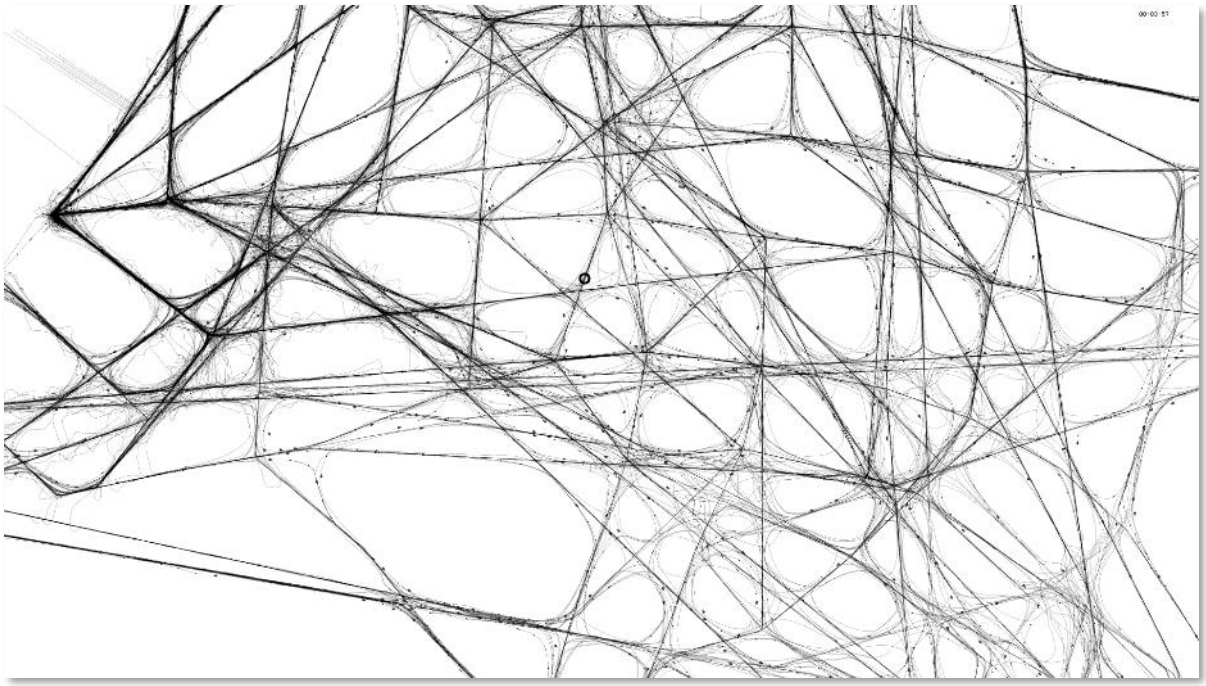


Figure 49 | Closer image of a set of agents conforming to the imported grid

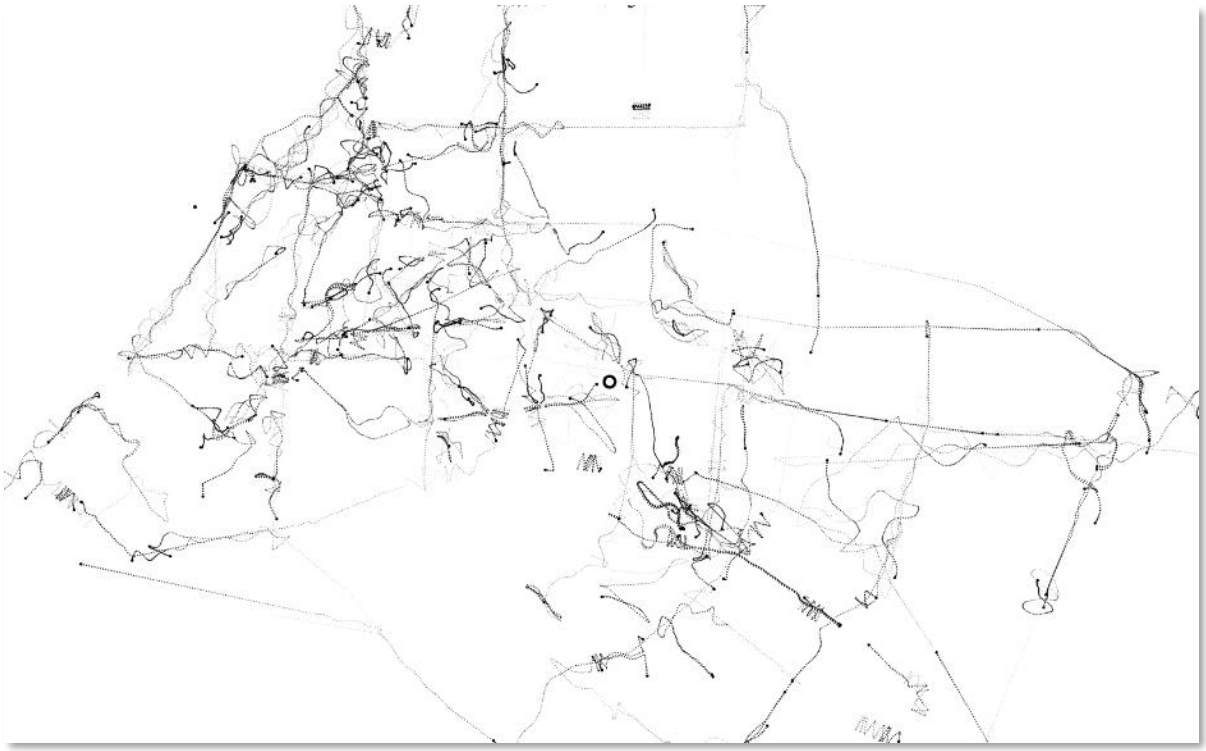


Figure 50 | Space negotiation between a set of agents and the conformed imported grid

Instead of creating agent sets on a flat surface, five agents were also created in high z position single (867 in this case) with the same exact rules as the zero z coordinate ones in order to create a three dimensional emergent shape set.

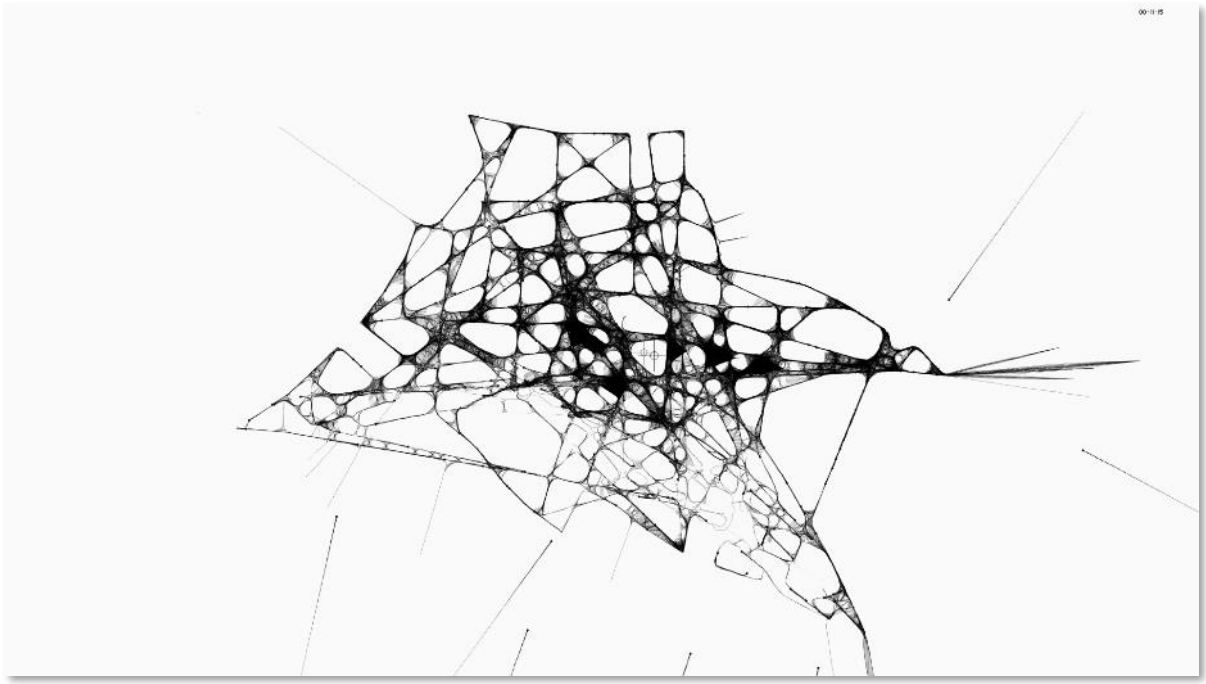


Figure 51 | Agents conforming to a grid with the line between tails function turned on.

In a third instance, single agents were placed with a high z value (867 in this case) with the same exact rules as the zero coordinate one in order to create a three dimensional emergent shape.

Because they were attracted to the rest of the flock and the grid itself they start to steer themselves down. As they are getting closer to the initial grid they start to approach it in an adaptive manner.

The way this system works allows us to work in various scales and create or own conditions for each simulation. It is always important to have in mind that each simulation is abstract and must be translated so it can have some interpretation. The same goes with its scale. One must adjust each variable for each test in order to get suitable results.

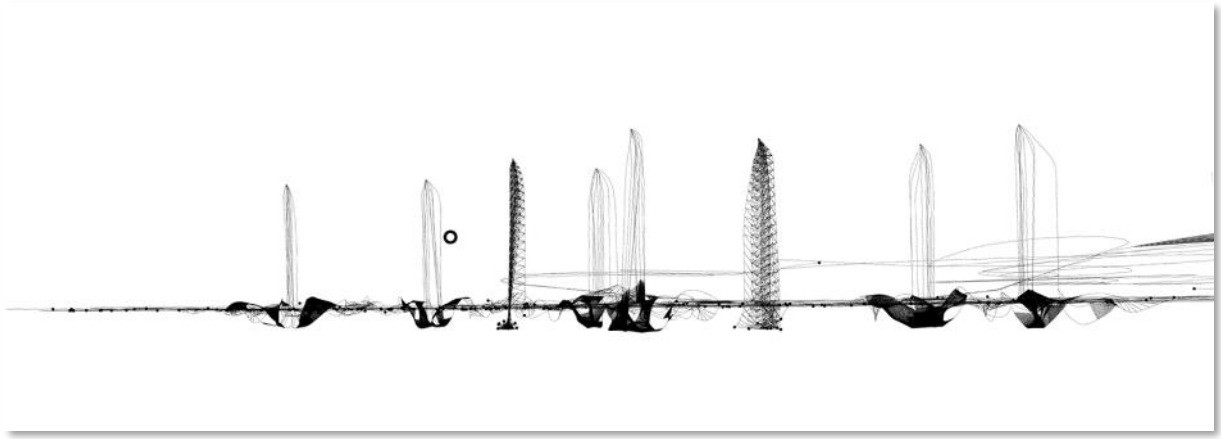


Figure 52 | Agents sets with three dimensional exploration.

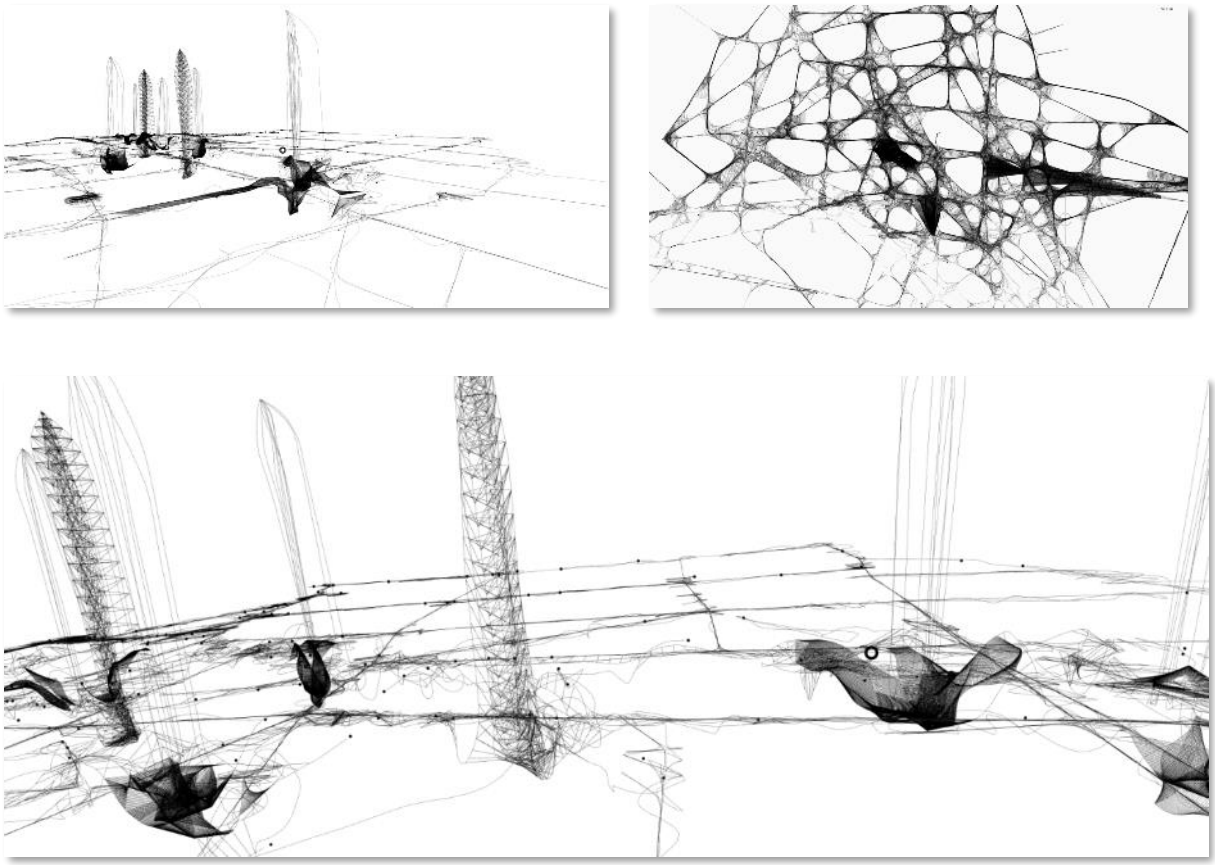


Figure 53 | Agents sets with three dimensional exploration(2).

4.4 Integrated Simulations on Context Specific Urban Environment

This section will address the integrated and context-specific simulation done in the development of this dissertation. They act as a final proof of concept for the application itself and also as a motivation for the implementation of certain functions that these specific simulations require namely, follow path, import external geometry, traffic implementation, Stigmergy and others.

4.4.1 Vila Nova de Cerveira-Goyan simulation

In sequence of the development of this dissertation arose the opportunity to combine the methods presented with other formal methods in a workshop environment. The main objective of the workshop was motivated by the intention of space analysis between Vila Nova de Cerveira and Goyan with the premise of a projection of a pedestrian bridge between two municipalities and on how urban fluxes could affect the most beneficial place for its placement. These two locations are connected by a bridge across the Minho River and have a profound socio-cultural connection.

Figure 54 | Trac(k)ing tracing by tracking workshop

Figure 55 | Map of the case study locations (Vila Nova de Cerveira / Goyan

Trac(k)ing tracing by tracking workshop described in this section was a suitable field test for the developed application as it condensed the main functions to achieve the intended features. Methods like importing an existing urban grid and defining behavioural looseness or tightness in terms of rule following were the main functions applied at this state. The collected data related to positive points and negative points that were then translated into attractors and deflectors inside the application.

The following simulations was part of a test made in the urban analysis workshop 'Trac(k)ing': tracing by tracking – a kinetic approach (planned methodology structured by Viana (2015)) in Escola Superior Gallaecia that embraced the articulation of the developed application with two other approaches focusing on studding the synergic relation between these two neighbour locations.

"'Trac(k)ing' tracing by tracking – a kinetic approach" was aligned by tree main methodologies, Space Syntax resolved by David Viana, Dynamic Mapping by Isabel Carvalho and Agent-based methods developed for this dissertation.

The objective of the workshop was to test and combine three different aproaches in a context-specific location, in this case Vila Nova de Cerveira and Goyan municipalities, while at the same time motivated by the intention of the prediction of a pedestrian bridge between these two locations. Although, mainly focused on how urban fluxes the workshop offered although some other useful like proximity relations information emerged from this exploration.

This turned into a case-study that focused on how the application could be used in an urban, real-world context in order to read space relations between two locations.

Figure 56 | Map of the imported geometry of studied area.

The workshop was planned into 4 main stages, first, took place a data collecting process through Dynamic Mapping, this was done by tracking several paths done by the workshop participants registered through GPS using a smartphone and a tracking application. In a second instance, the tracking data gathered was imported into the application. Lastly to run the application and extract the results to be articulated with the Space Syntax approach.

Figure 57 | Map of the imported geometry of studied area with roads.

The objective of the workshop was to test three different methodologies in a context-specific location, in this case Vila Nova de Cerveira and Goyan municipalities, while at the same time motivated by the intention of the prediction of a pedestrian bridge between two towns focuses on how urban fluxes could influence the decision for the most beneficial place for it although some other useful like proximity relations information emerged from this exploration.

Following the workshops structure:

The following simulations were part of a test made in the urbanism workshop 'Trac(k)ing': tracing by tracking – a kinetic approach (planned methodology structured by Viana (2015)) in Escola Superior Gallaecia that permitted the articulation of the developed application with two different approaches on studding the relation between two neighbour locations.

'Trac(k)ing' tracing by tracking – a kinetic approach was divided in to tree main methodologies, Space Syntax resolved by David Viana, Dynamic Mapping by Isabel Carvalho and Agent-Based Methodologies developed for this dissertation.

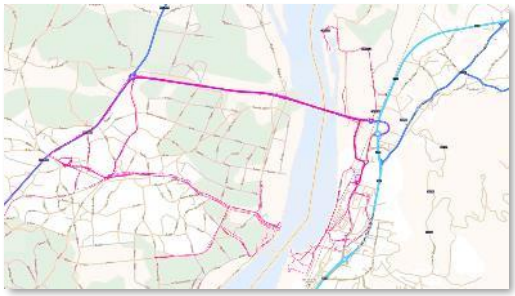


Figure 58 | Dynamic mapping example of the studied area.

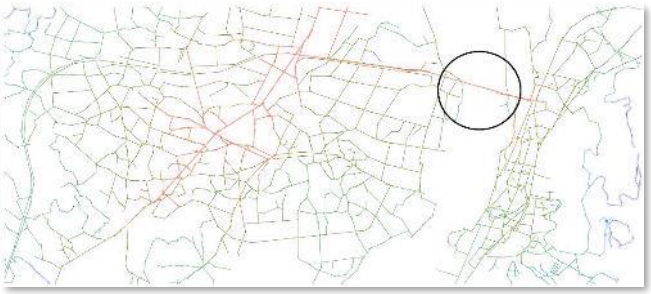


Figure 59 | Space syntax example of the studied area.

The synergy relations and dynamics between these two municipalities were the main features that were intended to be studied. These two towns are connected by a bridge across the Minho River and have a profound social connection. In this approach the question was how urban flux studies could influence the construction of a pedestrian bridge between the two and where could be the perfect location for that hypothetical bridge.

This turned into a case-study of how this Agent-Based methodologies application could be used in an urban, real-life application in order to read some intricate relations between two urban locations.

The workshop structure followed 4 main stages. First collecting data through the Dynamic Mapping method which cumulatively gathered all the positive and negative attractors, secondly to integrate the data produced into the developed application, to be able to run the simulation with, lastly to extract the results and compare them with the Space Syntax approach. This was done by tracing a hypothetical walk done by the workshop attendants and tutors (divided into groups) registered through GPS using a smartphone tracking application.

While doing this walk in and between these places each participant of each group would take a georeferenced photograph of specific places that would be of interest and sorting them into two types, positive places and negative places.

The next step was to implement the rules of attraction and deflection accordingly to the extracted data collecting into the ABM application. This would represent common positive and negative features of space based on first perception of space. The following phase was to spawn "createAgent" two sets of agents one in Vila Nova de Cerveira another in Goyan. For the sake of balance, agents were initially created in the city centre of each town and periodically through six entrance points, two in each town's centre and two in each location's main road entrances.

The attractors and deflectors mapped were then translated to the ABM application's coordinate system. In this way the translations between positive and negative places could make an influence and agents behaviour. On this stage the function "FollowRoads" was applied to prioritize the paths that have a road connection. On this case, because it's a larger scale, difference between agents that represent vehicles and pedestrians were differentiated. The difference between pedestrians and vehicles are the speed is that vehicles move on roads and they have a bigger speed than pedestrians. This means that the same agent ca be in one time a vehicle the other a pedestrian depending on the conditions rules and intentions. The strategy later applied was do determine which spot was the most obvious for agents to build the hypothetical bridge.

Figure 60 | Implementation of the positive and negative attractors into the studied location.

The problem with the early simulations was that agents were not aware that a bridge was going to be built somewhere between the two places so they walked around with no purpose other than reacting to attractors and deflectors revealing only the urban fluxes that existed.

At this stage a goal was implemented through a function according to a condition. If an agent were at one of the two centres of each town an intention would be activated to go the other following the rules that were defined to it.

A preliminary strategy was applied in a parallel and much more simplified simulation and consisted on distributing people along each river bank using the same intention to go to the other bank using attractors and deflectors. Although the results were consistent with the ones in the final simulation, these preliminary ones were highly biased since they worked as a proof of concept.

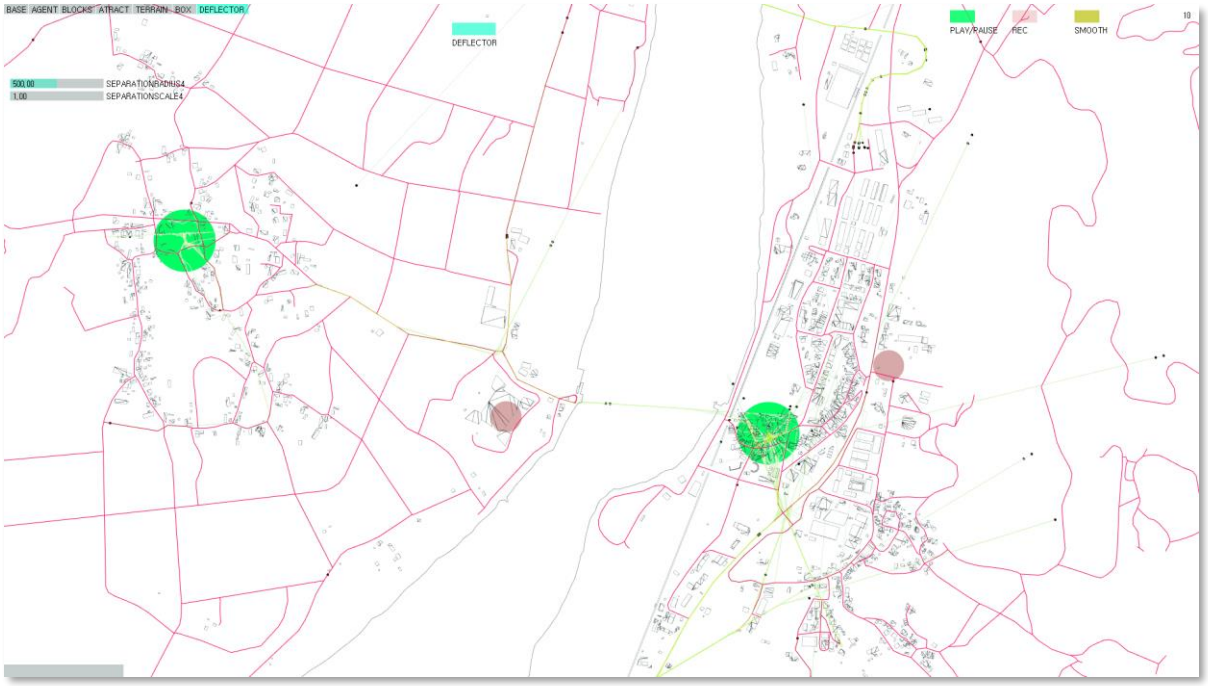


Figure 61 | First run of the workshop's set of simulations.

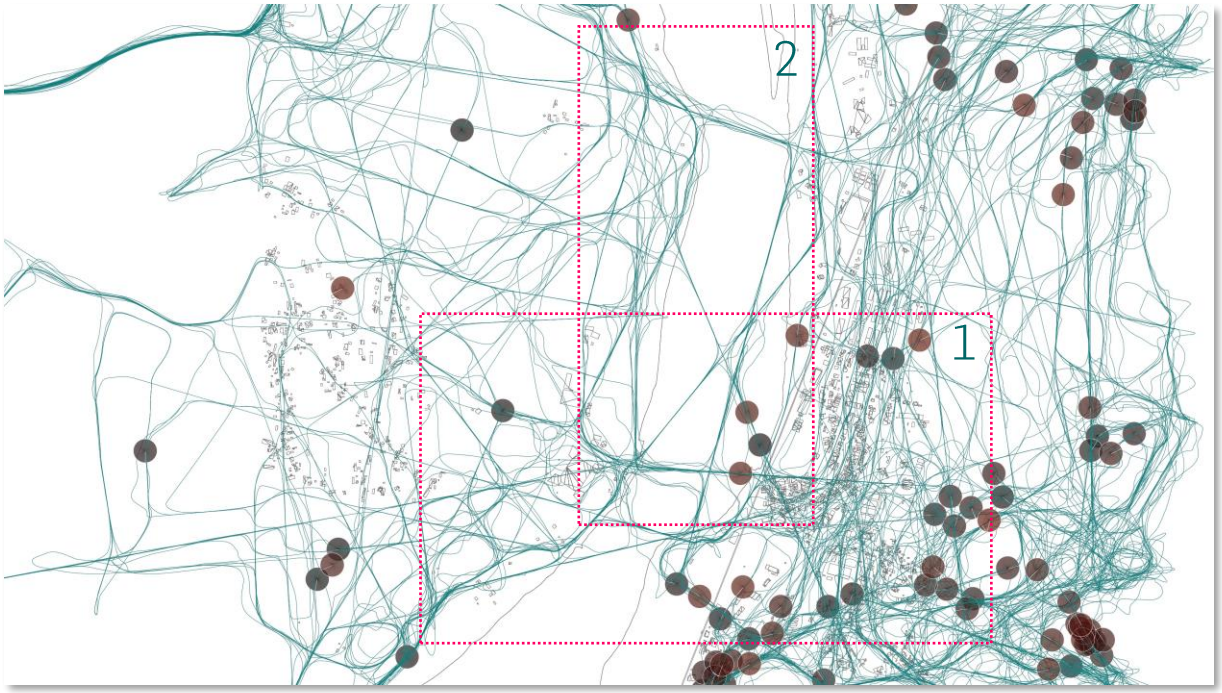


Figure 62 | Final results of the workshop

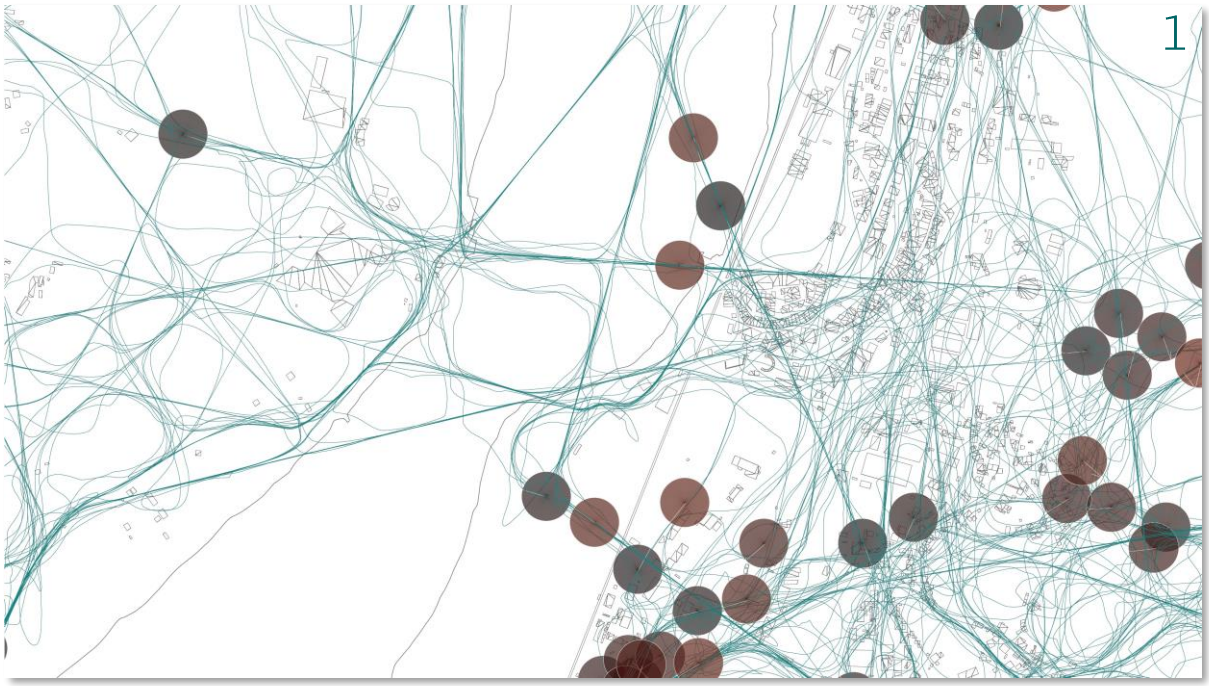


Figure 63 | Final results of the workshop (2)

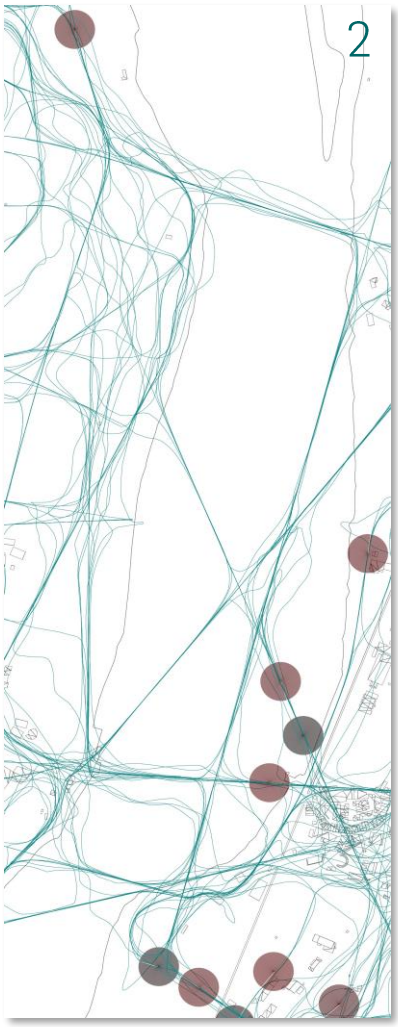


Figure 64 | Final results of the workshop (3)

The final approach was to run the final simulations and then correlate the results with the one that space syntax produced. Three position for the bridge were tested using this method and compared to the space syntax results with no pedestrian bridge only with the existing bridge.

In the agent-based method some unpredictable results are revealed for the hypothetical bridge. It is clear that the choice for the bridge is shifted towards the place where there is a smaller distance between the river banks.

When comparing the ABM results with the SS method, it is revealed that the choice for the location for the VNC bank is roughly around the same area although in neither of those cases, this location was the shortest line between banks.

Another relevant emergent result of this simulation was the fact that a two tendency path were generated along the GOYAN border in order to complete the lack of connections in this area.

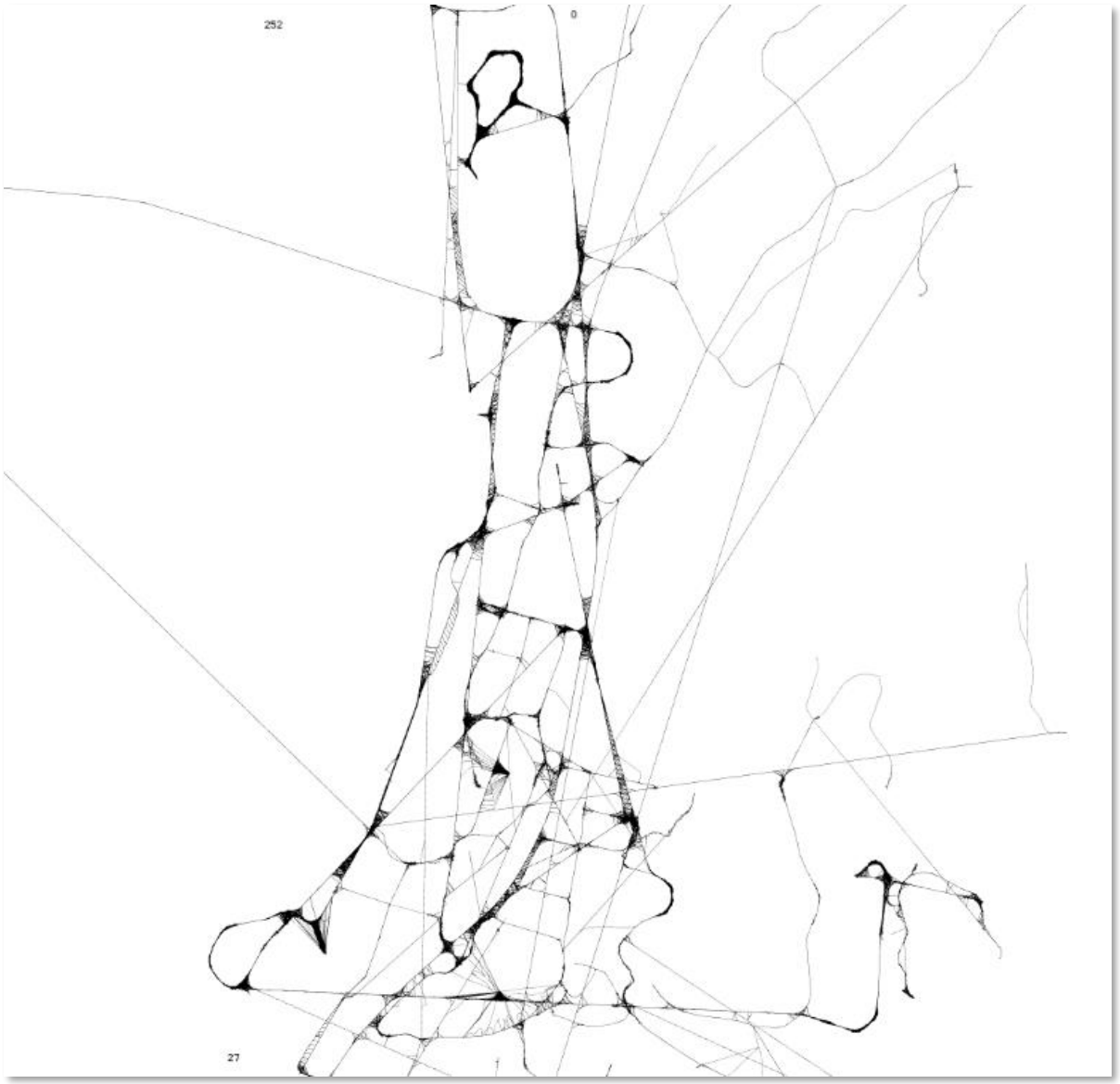


Figure 65 | Test on visualization methods (line between tails).

Although the results of this context specific simulation were clarifying on some particular concepts like flux and tendency paths, they didn't make the differentiation of pedestrian or vehicle, they are simply result of a cumulative set of simulations that were consistent to the predetermined lines of the imported grid. These representations sets showed specific uses for some of the functions integrated in the application like in the last example, a line between tails represented specific constrained spaces and curves that allowed for its recognition as such. These represent knots of highly desirable spaces that, spatially were not prepared for that amount of flux.

Following these assumptions, these issues revealed the need for a specific new set of rules that would deal with such conflict, so that the negotiation and cooperation between agent could be balanced and not strained. The next section embodies that need with the addition of another set of functions, pedestrian-vehicle differentiation, traffic lights to emulate traffic networks, and a

stigmergic algorithm to provide consistency and strengthen multi agent cooperation and extracted results.

4.4.2 Comprehensive Avenida dos Aliados simulation

As expressed in the previous section, the need of a differentiation between agents that symbolize pedestrians and agents that represented as vehicles was latent. This differentiation would provide a broader scale range for the simulations. The selection of a location for the context of the next set of simulations conformed to the following criteria:

Space configuration
The selected location is meant to have wide open spaces for pedestrians.
Must also have a transitional scale between the Batalha experiments and the Vila Nova de Cerveira – Goyan.
Element variety
Must have surrounding building that act as barriers.
Roads for vehicles with entrance and exit points.
Different potential buildings that could be translated to different types of attractors.
Entrance and exit points for pedestrians.
Accessibility to local Information
Information about the space (CAD plans, traffic light timings, etc...)
Possibility to be visited and observed in a first instance to collect input data.

Several possible locations met the above criteria although Av. Dos Aliados, Porto was the one selected as tested bed for the next series of implementations since it the intended features of the space were significantly explicit.

The first step was to recognize the relevant spatial features of this location which are directly related to the criteria chosen, attraction points, vehicle paths, entrance and exit points for both pedestrians and vehicles as well as traffic lights location and times.

Attractors were divided into two groups, dynamic and static. Dynamic meaning that they change position over time, and static that were fixed. Also the fixed attractors were divided into two groups, with different forces and influence radiuses. High radius, low radius, high force and low force.

Traffic was implemented according to three main entrance and exit points following specific path in and out and subjected to traffic light rules. This path is represented in blue in the second image of the next page. Note that these traffic agents have a particular set of values to meet the requirements of vehicle locomotion namely: no cohesion, high separation scale, low separation radius, high “FollowSpline” values, and recognition of traffic lights.

The pedestrian agents also have a specify set of values and functions that are tuned according to the test made in each simulation. The flock function operates as the previous simulations although, an extra separation function is implemented, this time, not only agents have a certain separation

among the same set, but also high separation values are implemented to make them separate from the vehicle agents. This provides a new level of interaction inside the system.

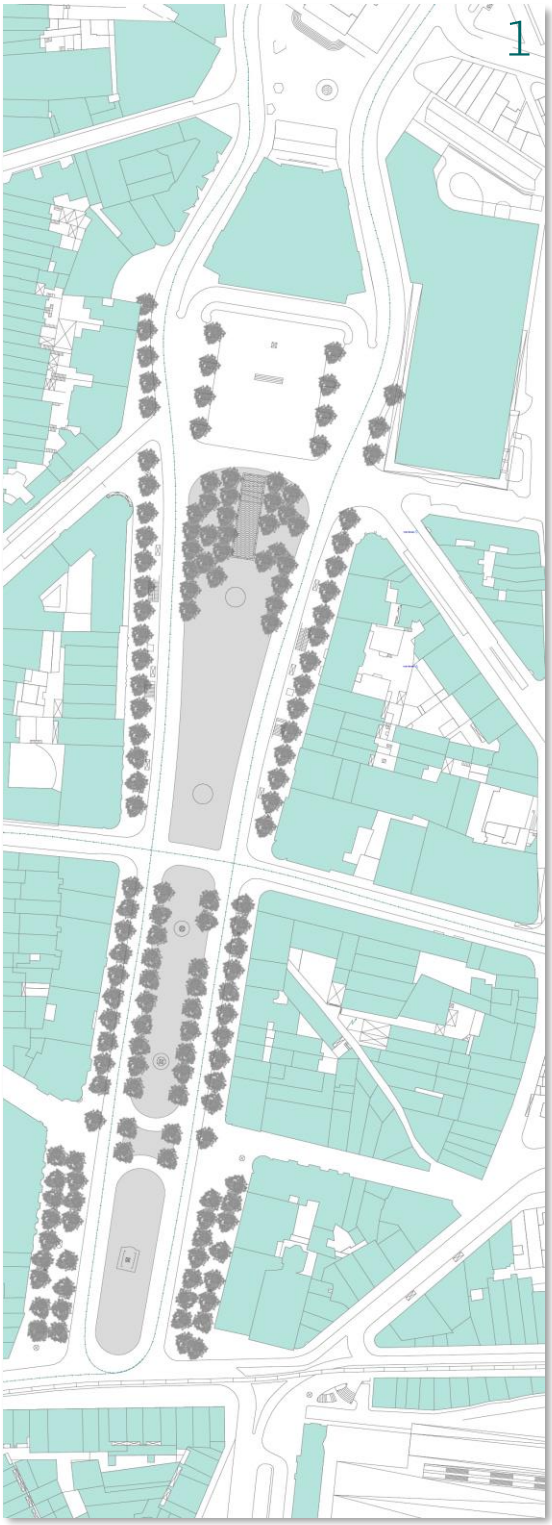


Figure 66 | Map of the imported geometry of studied area with buildings.

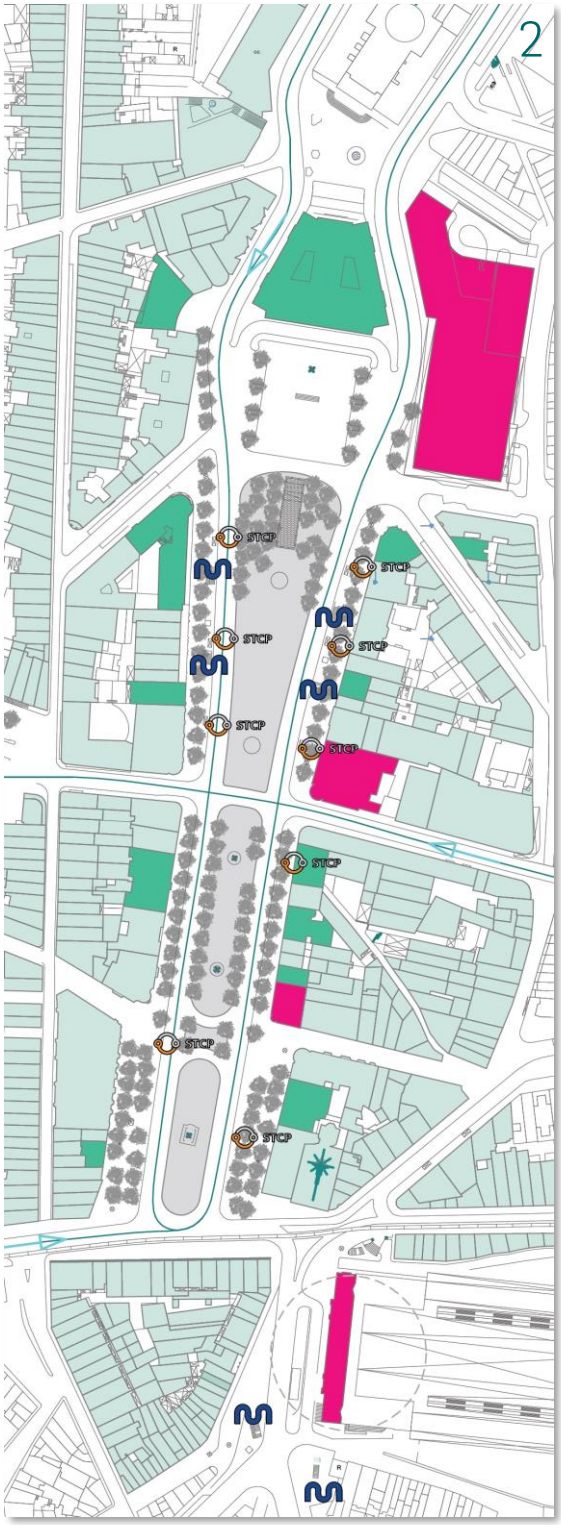


Figure 67 | Set of images of Avenida do Aliados, Porto, space configuration and spatial features.

Initial each of the images in the next page will be interpreted individually, and comparatively when required. The features that will be addressed and translated are flux, tendency paths and proximity interactions between agents and elements.

1 – The first image represents one of the early stage results and emphasizes the first set of pedestrian, agents represented as circles. This was a starting point to perceive how the basic rules are working. Rules like cohesion separation and alignment radiuses are represented in circles around each agent. This allow for an early stage adjustment of the variables and enables for the stabilization of the basic system. Values like maxSpeed and separation with buildings and vehicles are also adjusted in order to create a balanced system. What also can be seen in the first image is the tendency of all agents either to choose one side of the square or the other. This revealed that the amount of agents being created was not enough to cover all areas of this space so the agents set has the task of simply follow the attractor that is the closest.

2 – The second image reveals some of the constrains that happen when low separation values are being used. This type representation is achieved by using the function "Tail" with different colour and intensity values. The way it works is each agent leaves behind a red point at a specific rate, say, one each second and every time an agent gets away from this tail the previous one becomes slightly smaller and changes its colour incrementally ty green. This provides a type of reading that can expose features like a specific place being constrained or reveal desirable path in a short term.

This "Tail" function differs from the "Trail" function in a sense that, one represents short term paths and their nuances, other, a cumulative trace of the long path taken since the beginning of the simulation. This is evident in these two images

3 – In image three the previously revealed issues in 1 and 2 were adjusted in order to get a broader simulation. The number of agents was higher and with this higher number, it was able to fill the whole studied location. Another thing that can be visualized is the chaotic paths taken by the agents in constrained space. That does not happen in wide open spaces where some tendency paths start to emerge and be can perceived. These types of erratic behaviours led to an extensive adjustments of the variable in order to get relevant results in the next stages.

4 – This image represents the same exact results of image three but with added transparency to the buildings. This revealed some erratic behaviour by the agents which, in some cases, enter inside building. This was due to the fact that when agents were too close to a building and the current area was crowded with other agents, the last resort would be to get inside (a). This happens not only in this case but when an agent reaches a wall with a certain speed it ends up entering the building rather than go around it (b). This was an error that endured throughout the simulations that followed. Although a solution was found that avoided this type of behaviour, the performance of the application was lowered considerably losing the ability to adjust the variables in real-time. The erratic results were transformed from an error to a feature that allowed for the application to fulfil the relation between "inside and outside" and to determine what makes agents choose between these.

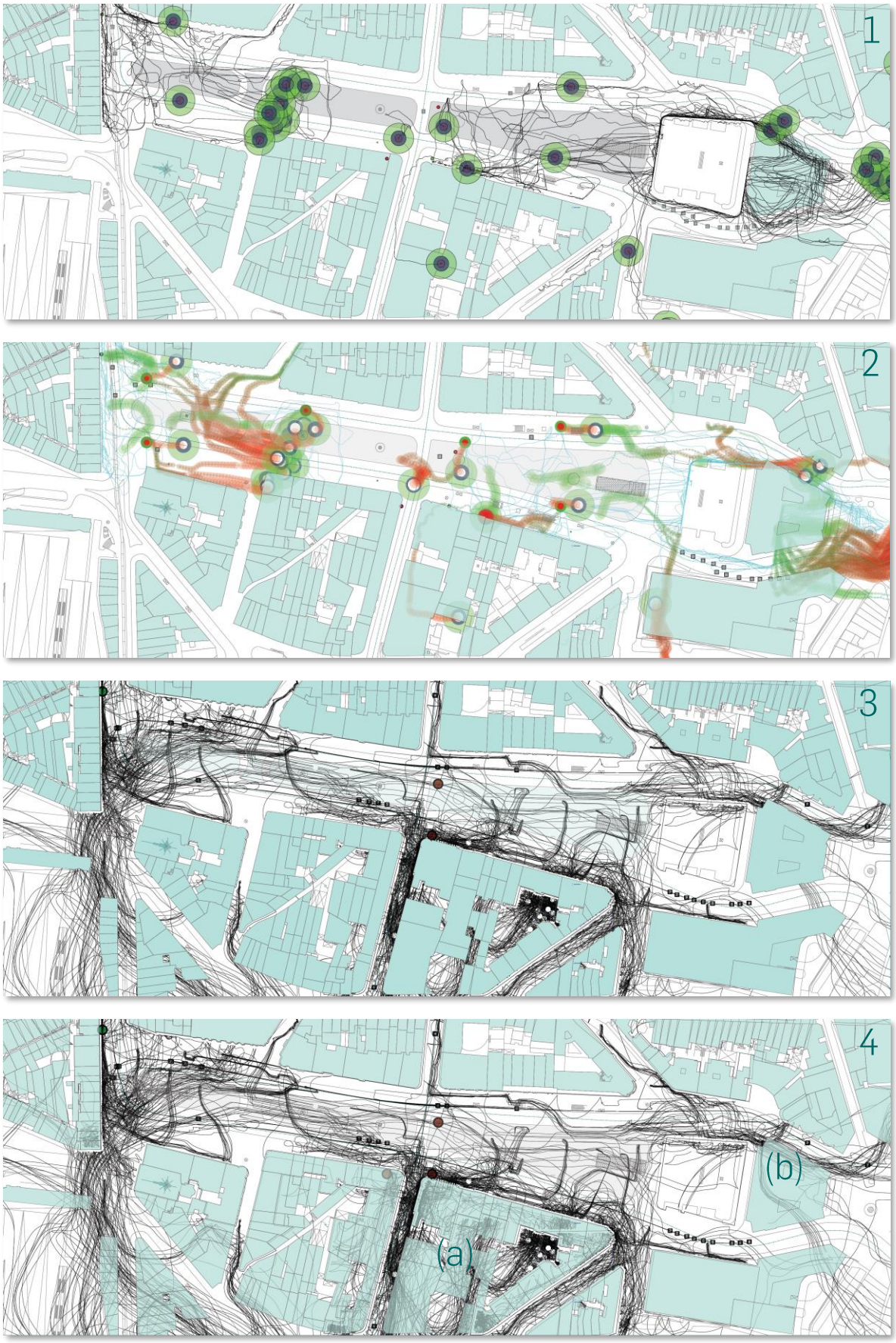


Figure 68 | Final results of Av. Dos Aliados simulations

The next images are the next iterations in the simulation and focus more on acquiring relevant graphical data of the location in question while still making use of certain adjustments of agent rules and functions.

5 – The image has some potentially relevant results, specifically, in relation to vehicle/pedestrian interaction (a). Pedestrian agents behave in a more directed way when confronted with traffic agents and consequently become confined to certain areas until that set of vehicles move away. At this stage the results revealed some deficiencies related to tendency path consistencies. Agent paths can be read individually but they lack of pronounced articulation among themselves, not allowing for the observation of patterns. This premise lead to the next series of adjustments.

6 – The goal that was mostly pursued in this dissertation was seeking tendency paths that were well pronounced and perceptible while at the same time reveal the influence of traffic forces in the agents. Although not really explicit, image 6, revealed a heavy influenced of traffic into the agents. Also, the variable that made a considerable difference goes back to the concept of scale addressed in chapter 2. The "maxSpeed" variable plays a major role in defining the scale in the simulation, nonetheless this scale only varies in terms of speed of the agents, the other variables and functions continue to have its own scale (cohesion scale, attraction scale, etc..). From this gap between scales returns a set of results that can be unbalanced and even biased while at the same time producing a set of results with features of its own. In summary, it always comes down to what features are being actually simulated and the more or less subjective translation of these features.

6 – Applying the principles of the tail function when the variables are stabilized results in another type of visualization that represents tai being dropped across the environment highlighting in red, zones that are more desirable than other, result of all the forces that are being applied attractors, both dynamic and static, traffic and flocking behaviours. At this point the results tend to reach a limit of their own capabilities, not only graphic wise but also in terms of features they represent. This limitation is mainly due to the fact that the simulation values are at a state of relative equilibrium, furthermore, not producing anything new and relevant to the results. What is concluded is that consistency, relevance and logical graphic representation was achieved.

Although that it is acknowledged that the simulation so far, produced results that achieved the premises based on the strategy guidelines, a certain lack of collaboration and self-awareness between agents is latent. This is what motivated the implementation of the naturally inspired concept of Stigmergy. At this stage, the agents and the results had to be slightly motivated to cooperate and explicitly expose a higher level of self-organization and awareness while at the same time revealing unmistakably the features being studied, specifically, tendency paths.

This conclusion led to the implementation of Stigmergy in the application and to address what it symbolizes when simulation human behaviour inside urban space. This method will be interpreted in the next pages (88-89) as a result of the previously addressed conclusions.

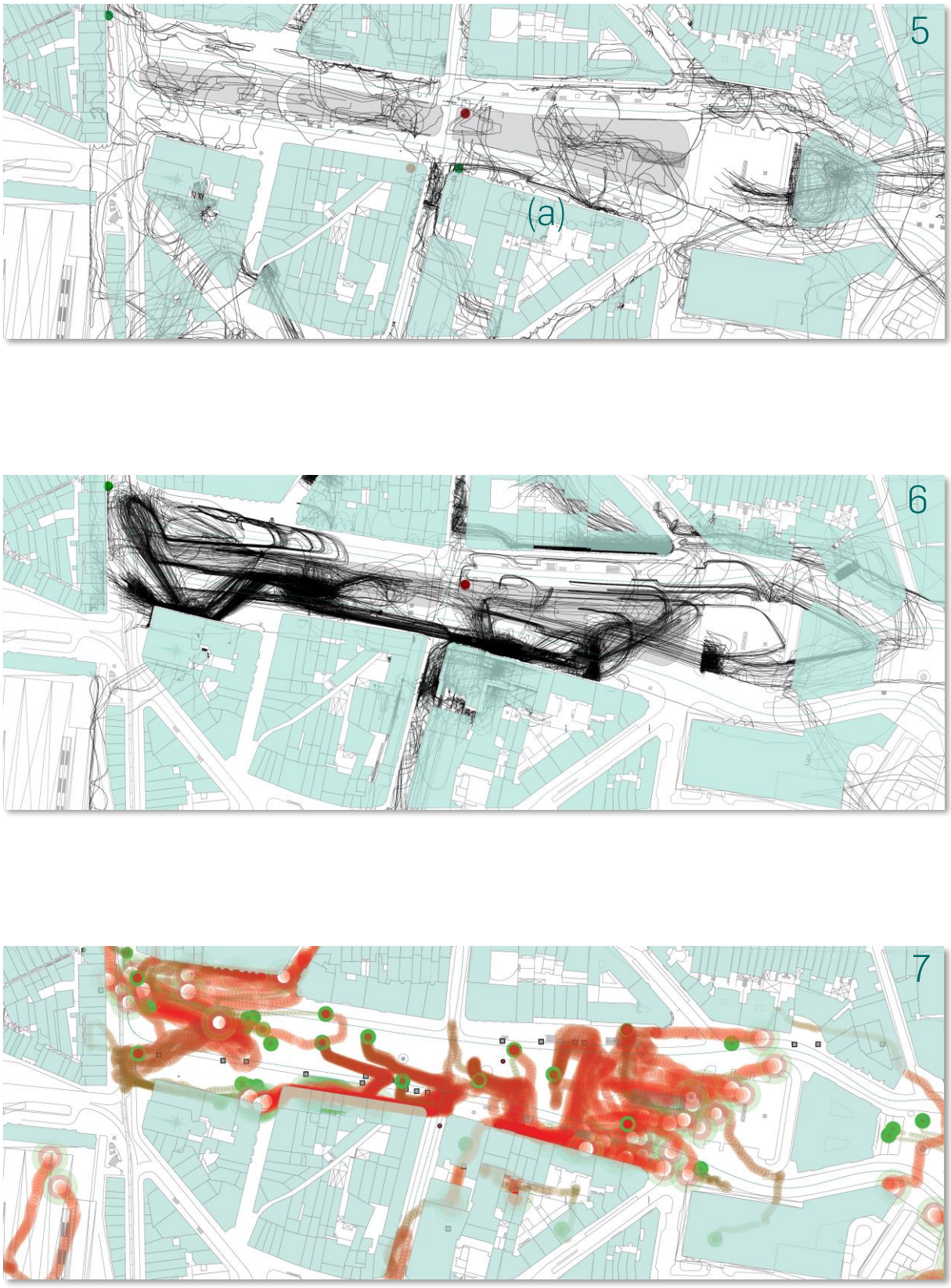


Figure 69 | Final results of Av. Dos Aliados simulations (2).

Stigmergy, in the next applied run of simulations, symbolizes the ability to perceive trails and either follow them or avoid them. As referred in the its assembly in chapter 6, this function acts as another decision maker for agent behaviour. Similarly, to what happened in the simulations that use the "followPath" function, agents start to follow paths that are present in their environment, the difference this time is that agents are creating their own paths and sharing them with other neighbour agents, this set of lines connected by points are named fields. This causes a dramatic emergent set of consequences, one of them is the ability to resolve an issue that happened in former simulations which is the absence of pronounced tendency paths.

This "shared memory" of paths that each agent leaves for others to seek, not only allows for agents to create much more defined tendency paths but also to gain a certain cooperation with a common goal. This goal is the result of all the other behaviours applied so far, (react to attractors, avoid vehicles, don't collide with buildings, go where others are but don't collide with them) but this time including the "follow paths that are known for their efficiency because someone else took them and they worked" behaviour. Although this quote is a rather simplistic explanation of how Stigmergy works, that's precisely what is happening when this function is triggered, agents start following paths that have a higher probability of being successful.

The features that resulted from the implementation of Sigmergy is explicit in the next set of images which are displayed with different types of graphic representations although they are result of the same simulation. In this image specifically, the field is shown as light green circles that fade after a certain amount of time. This fading means that after a certain amount of time the oldest field points start to disappear and consequently the less used paths begin to become less relevant until the simulation stabilizes revealing pronounced tendency paths.

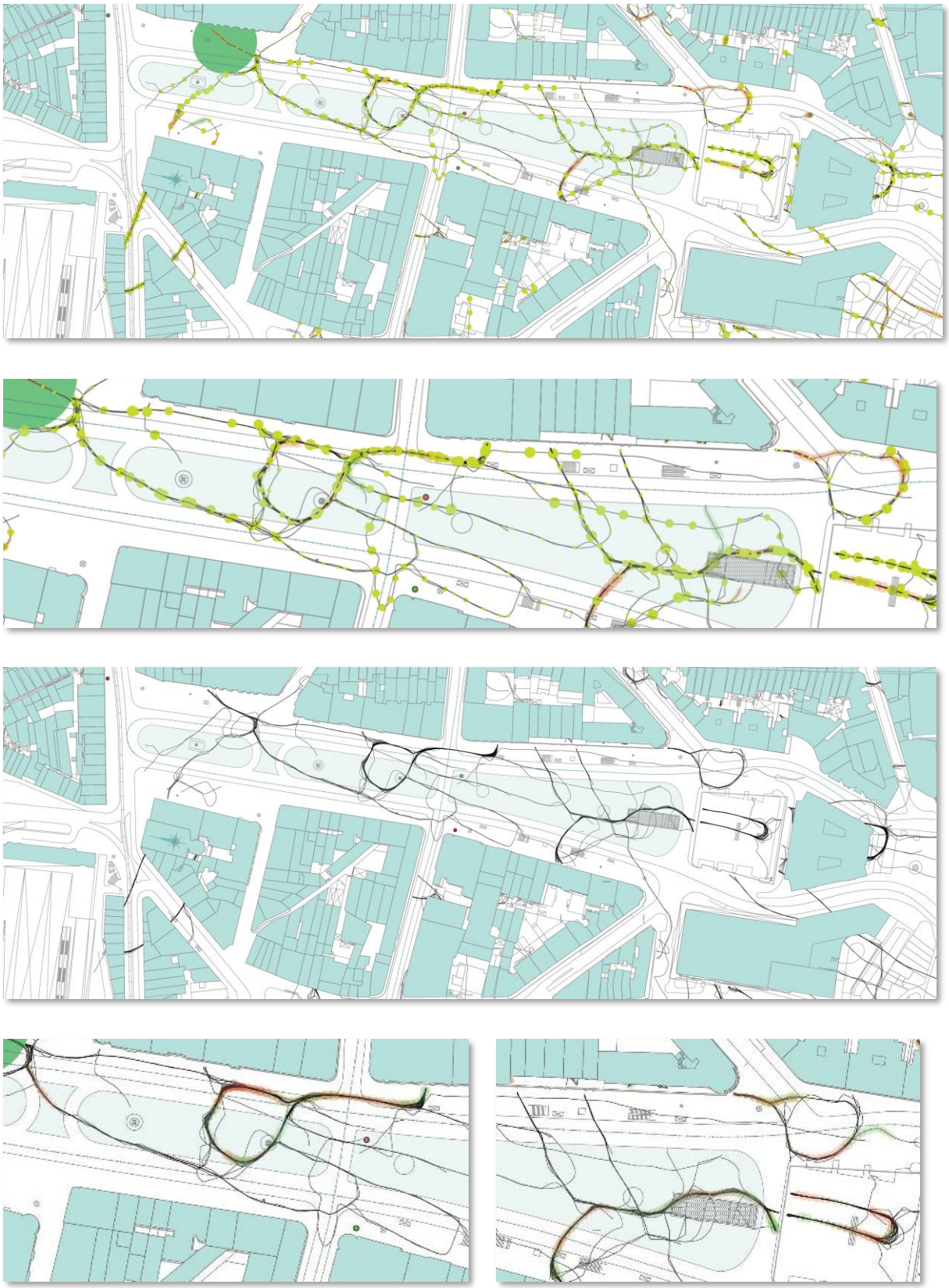


Figure 70 | Final results of Av. Dos Aliados with stigmergic behaviour.

The following list provides the potential advantages of integrating Stigmergy in the urban simulation process.

Advantages of Stigmergy
Improved Self-organization
Combined behaviour with attractors
Much more defined tendency paths which means more consistency throughout the simulations.
More "intelligence" in swarm intelligence.
Shared memory and cooperation between agents as opposed to action-reaction behaviour
Agents not only react to a specific attractor but also chose the most potentially successful path to that attractor and also create new ones that can be more or less used depending on their performance.
Every agent creates a path that can be followed for a certain amount of frames.
If the path created is chosen by other agents a number of times it becomes more and more consistent and thus, more used.
Stigmergy created a new spectrum of potential process that can be translated into ideas of social networks.
The graphic representation it produces, replicate a clearer way of recognizing patterns of human locomotion.
The cooperation between agent towards a common goal gains graphic consistency
Hierarchy between paths is created, more used paths, less used paths

This concludes the section of the integrated simulation results.

5 Result Correlation

This dissertation addresses the ability to use computational methods for dealing with the complexity of cities. It follows the knowledge explained by several authors of different fields to make use of agent based modelling and swarming in architectural urban contexts. This ability is defined in this dissertation in form of an application that can deal with the problem of interpreting and simulation large sets of information as it adopts a system and method that can adapt to emergent proprieties in the urban context. The results generated in the final stages of this dissertation were mostly directed towards the potential use of computation on spatial analysis and on social behaviour within urban environment, how these methods, while making use of natural locomotion processes, can be useful when simulation human locomotion inside a computer.

Although highly experimental, these methods can provide structured and consistent results that can be of use when approaching urban space depending on the quality and amount of input data to reveal specific proprieties of space. These proprieties and their definition are assumed as themes every simulation in this dissertation which are relevant to urban space analysis namely: (1) flux; (2) tendency paths and (3) openness/closeness features. These were the features that driven the development of the application, these capabilities were provided through a selection of naturally inspired models and their implementation.

Following from the beginning, the basic 1986 “boids” models from Reynolds through a Daniel Shiffman’s contemporary approach, the insights of Patrik Shummacher and Branko Kolarevic on the use of new technologies and computational processes in architecture, and the Robert Stuart Smith and Roland Snooks, approaches on context specific agent based methods, the results of this dissertation met the consistency it was intended to have in the search for a similar feature to what Christopher Alexander would call “quality without a name” and that otherwise would have turned into an uncontrollable bulk of meaningless imagery.

The central research question is: **How can emergence phenomena of human social behaviour in urban space can be simulated and logically represented graphically?**

As a response to this question this research developed a theoretical structure concerning Agent Based Modelling and Swarm Intelligence methodologies based on computational methods supported by a consistent selection of authors the refer these concepts. The main conception is based on the combination of Reynolds’s flocking behaviours and Swarm Intelligence’s theoretical model of Stigmergy, identifying these concepts as the first and last steps of this research, that ultimately met, the final stages of the software development, closed this research and allowed to, not only fulfil the pre-determined objectives, but also to provide moments of reflection on the results of each path taken throughout the whole process.

Conclusion

The computational approaches in this dissertation focused on providing a systemic process of **analysis of urban spaces through social behaviour computer simulations**, using **agent-based and swarm intelligence** methods by creating a process based on existing agent-based models and references to accommodate concepts like emergence and self-organization as well as simulation processes applied to specific urban contexts.

There was an underlining intention of creating a structure of tools that adapts to the needs of urban Architectural analysis in context-specific scenarios, allowing for them to be used in urban architectural practices. This intention responds to the early question: **How can these results be translated into a real-world scenario?** It is answered in the latter stages of this research where the developed model was applied to the two case studies, Vila Nova de Cerveira/Goián and Porto – Avenida dos Aliados.

As current modelling digital environments are trending to approach geometry in a more complex and dynamic way and continue to evolve in numerous fields - including architecture - the combination of computational tools of exploration increases and becomes more accessible to whomever needs to use them. This statement is a direct response to two of the questions that were constructed in the early stages of the development of this research which is: **Why should architecture theory embrace these types of methodologies?** And: **How are ABM and SI systems adapting to architecture procedures and how the architectural world uses them? And in what way?** This direct response to these questions is, in summary, these methods are mean to be an increment to the architectural knowledge by contributing with a set of tools that can provide different approaches when analysing urban architectural social behaviour.

Both objectives of this research were fulfilled by providing a constantly developing tool that uses the potential of agent based and swarming models for simulation and urban space analysis and perceive emergent proprieties within urban and social space that are complex by nature as well as to review its potential in the integration with architecture's cultural techniques and practices.

Although the possibilities of this agent based modelling to explore context specific spatial features can be relevant in an abstract way they are always subject to interpretation and translation to concepts that are used in a real-world context. In order to fill this gap between these two different worlds, the future of this research will focus on providing a foundation for the creation of a common language that will make this translation continuous. This answers one of the questions defined at an early stage of this research which is: **How can this graphic information can be interpreted?** This leads to the conclusion that most of these results are highly volatile and this interpretation is one of the fundamental elements of the developed process. In sum, the results extracted are deeply dependant on this translation of which elements represent which features of space and how they can be validated.

Agent-based and Swarm Intelligence models were the support of this dissertation and empowered the potential knowledge of how cities operate as complex systems, in terms of spatial organization and social behaviour embodying natural rules of motion into a computer application.

Although Stigmergy was produced one of the most significant results in this dissertation, this method is still at an early stage and it's meant to be developed in the future in order to improve the application performance.

Finally, this dissertation represents a first step towards the continuous development of an application that is meant to provide a toolset to fulfil a specific type of urban analysis as agent-based and swarm intelligence models show a potential to reveal, still hard to reach, features of urban space. This also represents a contribution to the cultural techniques inside the architectural r e s e a r c h f i e l d . o k

References

Alexander, C. (1977). *A pattern Language: Towns, Buildings, Construction.* New York: Oxford University Press.

Beirão, J. N. (2012). CityMaker: Designing Grammars for Urban Design. *Architecture and built Environment,* 55 1-272. Recovered from http:// abe.tudelft.nl/index.php/faculty-architecture/article/download/beirao/beirao

Bonabeau, E., Dorigo, M., & Theaulaz, G. (1999). *Swarm Intelligence: From Natural to Artificial Systems.* New York: Oxford University Press.

Carpo, M. (2013). *Introduction: Twenty Years of Digital Design, in Digital Turn in Architecture 1992-2012.* Chinchester: Wiley.

Corner, J. J., & Lamont, G. B. (2004). Parallel Simulations of UAV Swarm Scenarios. *Proceedings of the 2004 Winter Simulation Conference. Hunting Beach, California.* pp. 355-63. R. G. Ingalls et al.

Deleuze, G., & Guattari, F. (1987). *A Thousand Plateaus.* Minneapolis: University of Minnesota Press.

ESAP. (2015, November). *Formal Methods in Architecture*. Retrieved from Formal Methods in Architecture. Recovered from http:// archformalmethods.wixsite.com

Hardt, M., & Negri, A. (2004). *Multitude: War and Democracy in the Age of Empire* . New York: Penguin.

Johnson, S. (2001). *Emergence: The Connected Lives of Ants, Cities and Software.* New York: Scribner.

Kolarevic, B. (2015). *Building Dynamics, Exploring Architecture of Change.* New York: Routledge, Taylor & Francis Group.

Oosterhuis, K., & Feireiss, L. (2006). *Game, Set and Match II. On Computer Games, Advanced Geometries, and Digital Technologies.* Rotterdam: Episode Publishers.

Reynolds, C. W. (1996). *Steering Behaviours For Autonomous Characters.* California: Sony Computer Entertainment America.

Reynolds, C. W. (1996). *Steering Behaviours for Autonomus Characters.* Recovered from http://www.red3d.com/cwr/papers/1999/gdc99steer.pdf

Schumacher, P. (2010). *The Autopoiesis of Architecture, A New Framework for Architecture, (Vol. 1),* Chinchester: John Wiley & Sons. ISBN 978-0-470-77298-0

Shiffman, D. (2012). *The Nature of Code - Simulating Natural Systems with Processing.* self-published. ISBN-13 978-0985930806

Snooks, R. (2014, April 01). suckerPUNCH: Interview with Roland Snooks. (suckerPUNCH, Interviewer)

Snooks, R., & Stuart-Smith, R. (2016). Kokkugia : Recovered from *http://www.kokkugia.com/about*.

Valckenaers, P., Kollingbaum, M., Van Brussel, H., Bochmann, O., & Zamfirescu, C. (2016). *THE DESIGN OF MULTI-AGENT COORDINATION AND CONTROL SYSTEMS USING.* Belgium: Katholieke Universiteit Leuven.

Vehlken, S. (2014). Computational Swarming: A Cultural Technique for Generative Architecture. *FOOTPRINT Delft Architecture Journal, 15,9-14. DOI: 808-1104-1-PB*

Vogl, J. (2001). Medien-Werden: Galilieos Fernrohr, pp. 115-123. Recovered from: https://www.fink.de/katalog/reihe/archiv_fuer_mediengeschichte.html

Index of tables and figures